

Evaluation of Security Issues on Communication Systems for Internet of Things (IoT)

Mostafa Ayoubi Mobarhan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
March 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer
Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Mohammed Salamah
Supervisor

Examining Committee

1. Prof. Dr. Doğu Arifler

2. Prof. Dr. Semih Bilgen

3. Prof. Dr. Mohammed Salamah

4. Prof. Dr. Emrullah Turhan Tunalı

5. Assoc. Prof. Dr. Gürcü Öz

ABSTRACT

In this thesis, taking into consideration of performance-based and security issues, we suggest a set of secure and lightweight AKA methods for LTE network and LTE-enabled IoT applications named Robust Evolved Packet System-Authentication and Key Agreement (REPS-AKA) protocols. Accordingly, we presented five security approaches called REPS-AKA1, REPS-AKA2, REPS-AKA3, REPS-AKA4, and REPS-AKA5. The REPS-AKA1 protocol is presented in chapter 3 to satisfy the security requirements of the LTE system. Then, new handover mitigation named REPS-AKA2 is also suggested in chapter 4 to provide backward and forward key separation and satisfy important security requirements in the X2 handover. In chapter 5, the REPS-AKA3 protocol is presented to settle the security issues in LTE networks. The suggested approach combines the asymmetric and symmetric encryption methods to provide a strong mutual authentication between involved nodes. In chapter 6, we proposed REPS-AKA4 method as a set of safe and fast authentication protocols during the first attachment, n^{th} attachment, and Inter/Intra handover in LTE-WLAN Interworking (LWI) systems. In addition, we suggested a lightweight group-based AKA method for IoT applications called REPS-AKA5 protocol in chapter 7. The proposed method presents a strong mutual authentication between involved nodes and ensures strong identity protection of the IoT devices. The suggested approach also presents considerable confidentiality in the user/control planes using an optimized combination of symmetric/asymmetric keys. We verified the proposed AKA protocols by using the AVISPA, Scyther tools, and BAN logic to check that the REPS-AKA methods are resistant to different known security attacks. Furthermore, we evaluated our method based on performance metrics. The considered results are shown that compared with

other AKA methods, the proposed methods provide better performance in terms of computational cost, and signaling overhead. Hence, the REPS-AKA approaches are able to achieve not only defined security requirements but also a considerable improvement in network performance.

Keywords: M2M communication, LTE Security, AVISPA tool, BAN logic, Group-based authentication, IoT systems, Subscriber identity, and EPS-AKA protocol, Mobile Communication, Mutual Authentication.

ÖZ

Bu tezde, performansa dayalı ve güvenlik konularını dikkate alarak, LTE ağı ve LTE özellikli IoT uygulamaları için Robust Evolved Packet System-Authentication and Key Agreement (REPS-AKA) adı verilen bir dizi güvenli ve hafif AKA yöntemi öneriyoruz. Buna göre, REPS-AKA1, REPS-AKA2, REPS-AKA3, REPS-AKA4, ve REPS-AKA5 adı verilen beş güvenlik yaklaşımı sunuyoruz. Bölüm 3'te, LTE sisteminin güvenlik gereksinimlerini karşılamak için REPS-AKA1 protokolü sunulmuştur. Daha sonra, yeni geçiş işlemi adı verilen REPS-AKA2, geriye ve ileriye doğru anahtar ayrımı sağlamak ve X2 geçişinde önemli güvenlik gereksinimlerini karşılamak için Bölüm 4'te önerilmiştir. 5. bölümde, REPS-AKA3 protokolü LTE ağlarındaki güvenlik sorunlarını çözmek için sunulmuştur. Önerilen yaklaşım, ilgili cihazlar arasında güçlü bir karşılıklı kimlik doğrulama sağlamak için ortak anahtar ve simetrik anahtar yöntemlerini birleştirir. 6. Bölümde, LTE-WLAN sistemlerinde ilk bağlantı, n'inci bağlantı ve inter/intra aktarım sırasında bir dizi güvenli ve hızlı doğrulama metodu olarak REPS-AKA4 protokolünü önerdik. Önerilen REPS-AKA4 protokolü, yalnızca güvenli bir inter/intra kimlik doğrulaması değil, aynı zamanda AKA prosedürünün performansını arttırabilen LWI sistemi içerisinde etkin bir yeniden doğrulama yaklaşımı sağlar. Ek olarak, 7. bölümde REPS-AKA5 protokolü olarak adlandırılan IoT uygulamaları için hafif grup-tabanlı bir AKA yöntemi önerdik. Önerilen yöntem, ilgili düğümler arasında güçlü bir karşılıklı kimlik doğrulama sunar ve IoT cihazları için güçlü kimlik koruması sağlar. Önerilen yaklaşım aynı zamanda, optimize edilmiş bir simetrik/asimetrik anahtar kombinasyonu kullanan kullanıcı/control planları için önemli ölçüde gizlilik sunar. REPS-AKA yöntemlerinin bilinen farklı güvenlik saldırılarına karşı dirençli olduğunu kontrol etmek için

AVISPA, Scyther araçları ve BAN mantığını kullanarak önerilen AKA protokollerini doğruladık. Ayrıca, yöntemimizi performans metriklerine göre değerlendirdik. Elde alınan sonuçlar, diğer AKA yöntemleriyle karşılaştırıldığında, önerilen yöntemlerin hesaplama maliyeti, ve sinyal yükü açısından daha iyi performans sağladığını göstermektedir. Bu nedenle, REPS-AKA yaklaşımları, yalnızca tanımlanmış güvenlik gereksinimlerine ulaşmakla kalmaz, aynı zamanda ağ performansında da önemli bir gelişme sağlar.

Anahtar Kelimeler: Makineden Makineye İletişim (M2M), LTE Güvenliği, AVISPA aracı, BAN mantığı, Grup-tabanlı kimlik doğrulama, IoT sistemleri, Abone kimliği, EPS-AKA protokolü, Mobil iletişim, Karşılıklı kimlik doğrulama.

DEDICATION

I would like to dedicate my work to my lovely wife, father, and mom.

ACKNOWLEDGMENT

I want to show my sincere appreciation to Prof. Dr. MOHAMMED SALAMAH for his supervision, kind advice, and management from the beginning of my thesis work till this stage. He was my guiding mentor who did not hesitate even for a second to keep enriching me with his extraordinary experiences throughout the work. He played an essential role in being a psychological guide, ensuring that I do not lose this enthusiasm, especially during the publication stage. He was an easy-going proficient, a teacher, and a friend. His encouragement and support were endless and a true inspiration to think outside the box. I owe him a lot that even words are not enough to show him gratitude.

I also want to show appreciation to the department chair, Prof. Dr. ZEKİ BAYRAM, and vice-chairs, Prof. Dr. MOHAMMED SALAMAH, and Assist. Prof. Dr. AHMET ÜNVEREN for their kind-heartedness and their generosity. Also, I want to show gratitude to the monitoring committee members for their helpful advice and assistance, especially Prof. Dr. DOĞU ARIFLER for supporting and guiding me without any regrets. Also, not to forget to mention Assoc. Prof. Dr. GURCU ÖZ, for her involvements. I thank them all for their significant contribution and comments that added great quality to my research work.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	v
DEDICATION.....	vii
ACKNOWLEDGMENT	viii
LIST OF TABLES.....	xiv
LIST OF FIGURES	xvii
LIST OF SYMBOLS AND ABBREVIATIONS	xvix
1 INTRODUCTION	1
1.1 Architecture of LTE network	2
1.2 Security architecture of LTE system	4
1.3 Security loopholes of LTE network	7
1.4 Thesis contribution	10
2 LITERATURE REVIEW	12
2.1 Simulations and security tools	12
2.1.1 AVISPA verification tool	12
2.1.2 Scyther verification tool	14
2.1.3 BAN logic	16
2.2 Related works of LTE protocols	19
2.3 Related works of group-based protocols	20
3 MODEL 1 (REPS-AKA1)	23
3.1 The proposed REPS-AKA1 protocol.....	23
3.2 Security analysis of REPS-AKA1 protocol.....	30
3.2.1 Formal analysis using AVISPA.....	30

3.2.2 Formal analysis using Scyther	32
3.2.3 Security analysis by BAN logic.....	35
3.2.4 Proof of security requirements.....	37
3.3 Performance analysis of the proposed model: REPS-AKA1.....	42
3.3.1 Bandwidth consumption (core traffic).....	42
3.3.2 Computational cost	45
3.3.3 Signaling overhead	47
3.3.4 Storage overhead in the serving network.....	48
4 MODEL 2 (REPS-AKA2).....	50
4.1 The proposed REPS-AKA2 protocol	50
4.1.1 Phase1 (Ph1: Initial Authentication).....	51
4.1.2 Phase2 (Ph2: re-Authentication).....	54
4.2 Security analysis of REPS-AKA2 protocol	56
4.2.1 Formal analysis using AVISPA.....	57
4.2.2 Formal analysis using Scyther	58
4.2.3 Security analysis by BAN logic.....	61
4.2.4 Proof of security requirements.....	66
4.3 Performance analysis of the proposed model: REPS-AKA2.....	72
4.3.1 Communication overhead	72
4.3.2 Transmitted packets	73
4.3.3 Computational cost	74
5 MODEL 3 (REPS-AKA3)	75
5.1 The proposed REPS-AKA3 protocol	75
5.1.1 Pre-authentication phase	75
5.1.2 EPS security context	78

5.1.3 NAS Security context	83
5.1.4 AS Security context	84
5.1.5 Re-authentication procedure	86
5.2 Security analysis of REPS-AKA3 protocol	88
5.2.1 Formal analysis using AVISPA	88
5.2.2 Formal analysis using Scyther	90
5.2.3 Security analysis by BAN logic	92
5.2.4 Proof of security requirements	97
5.3 Performance analysis of the proposed model: REPS-AKA3	105
5.3.1 Bandwidth consumption (core traffic)	105
5.3.2 Bandwidth consumption (total traffic)	108
5.3.3 Computational cost	110
5.3.4 Signaling overhead	112
5.3.5 Storage overhead in the serving network	113
6 MODEL 4 (REPS-AKA4)	115
6.1 LTE-WLAN systems	117
6.2 The proposed REPS-AKA4 protocol	118
6.2.1 REPS-AKA4 Ph.1.1	119
6.2.2 REPS-AKA4 Ph.1.2	122
6.2.3 REPS-AKA4 Ph.2.1	124
6.2.4 REPS-AKA4 Ph.2.2.1	127
6.2.5 REPS-AKA4 Ph.2.2.2	130
6.2.6 REPS-AKA4 Ph.3.1	133
6.2.7 REPS-AKA4 Ph.3.2	135
6.2.8 REPS-AKA4 Ph.4.1	137

6.2.9 REPS-AKA4 Ph.4.2	139
6.3 Security analysis of REPS-AKA4 protocol.....	142
6.3.1 Formal analysis using AVISPA.....	142
6.3.2 Formal analysis using Scyther	143
6.3.3 Security analysis by BAN logic.....	152
6.3.4 Proof of security requirements.....	161
6.4 Performance analysis of the proposed model: REPS-AKA4.....	164
6.4.1 Evaluation of LTE-AKA methods (REPS-AKA4 Ph.1.1/2)	165
6.4.1.1 Bandwidth consumption (core traffic).....	165
6.4.1.2 Bandwidth consumption (total traffic)	168
6.4.1.3 Computational cost	169
6.4.2 Evaluation of LWI-AKA methods	172
6.4.2.1 Authentication delay.....	173
6.4.2.2 Signaling cost.....	177
7 MODEL 5 (REPS-AKA5).....	180
7.1 Architecture of LTE-enabled IoT system	183
7.2 The proposed REPS-AKA5 protocol.....	185
7.2.1 Group construction phase	186
7.2.2 First authentication phase	187
7.2.3 Re-authentication phase.....	191
7.2.3.1 Complete re-authentication.....	191
7.2.3.2 External re-authentication.....	194
7.2.3.3 Internal re-authentication.....	196
7.2.4 Group updating phase	197
7.2.4.1 Group leader/backup leader updates.....	198

7.2.4.2 Group member updates.....	199
7.3 Security analysis of REPS-AKA5 protocol.....	201
7.3.1 Formal analysis using AVISPA.....	201
7.3.2 Formal analysis using Scyther	202
7.3.3 Security analysis by BAN logic.....	205
7.3.4 Proof of security requirements.....	209
7.4 Performance analysis of the proposed model: REPS-AKA5.....	220
7.4.1 Total bandwidth consumption	221
7.4.2 Bandwidth consumption (core traffic).....	225
7.4.3 Computational cost	228
7.4.4 Signaling overhead	233
7.4.5 Storage overhead in the serving network.....	235
8 CONCLUSION	237
REFERENCES	244
APPENDICES	263
Appendix 1 : AVISPA codes	264
Appendix 2 : Scyther codes	274

LIST OF TABLES

Table 1: Notations of BAN Logic	17
Table 2: Cryptography functions in REPS-AKA1	26
Table 3: Security goals in REPS-AKA1.....	31
Table 4: Evaluation of the presented AKA methods	42
Table 5: Assumption values for Traffic's model.....	42
Table 6: Comparison of SO metric	48
Table 7: The <i>StO</i> of different AKA protocols	48
Table 8: Security goals in REPS-AKA2.....	57
Table 9: Description of message authentication in REPS-AKA2	68
Table 10: Asymmetric and symmetric keys in REPS-AKA2.....	69
Table 11: Comparison of security goals in Handover methods.....	71
Table 12: Comparison of security properties in Handover schemes	72
Table 13: Comparison of communication overheads	73
Table 14: Computation times of involved nodes	74
Table 15: Example of AAT	76
Table 16: Example of FDT	77
Table 17: Example of ST.....	77
Table 18: Example of EIT	78
Table 19: Cryptographic functions	79
Table 20: Security goals of REPS-AKA3	88
Table 21: Asymmetric and symmetric keys	100
Table 22: Message authentication in REPS-AKA3.....	102
Table 23: Evaluation of security features in AKA methods.....	104

Table 24: Comparison of security goals in AKA protocols	104
Table 25: Assumption values.....	105
Table 26: Transmitted bits in AKA protocols	109
Table 27: Comparison of signaling overhead.....	113
Table 28: Storage overhead of AKA methods.....	113
Table 29: Asymmetric and symmetric encryption in REPS-AKA4.....	164
Table 30: The assumption values	165
Table 31: Authentication delay in different scenarios.....	176
Table 32: <i>NTM</i> and <i>AMS</i> values of AKA protocols	177
Table 33: Message authentication in REPS-AKA5.....	210
Table 34: Asymmetric and symmetric keys in REPS-AKA5.....	216
Table 35: Comparison of security properties in Group-based AKA protocols	220
Table 36: Security comparison of the Group-based AKA protocols.....	220
Table 37: Size of the used parameters in the REPS-AKA5.....	221
Table 38: The <i>TBC</i> (in Bits) of group based AKA protocols (per message).....	222
Table 39: The ratio of <i>TBC</i> for different AKA protocols (when $m=50$)	223
Table 40: The average improvements (TBC) of the REPS-AKA5	224
Table 41: Assumption values (traffic flow).....	225
Table 42: Cost (Time delay) of the different operations	231
Table 43: Total delay time of the different AKA protocols	232
Table 44: Signaling overhead of the AKA protocols	234
Table 45: Storage overhead of AKA methods.....	235
Table 46: Storage overhead of group-based protocols	236

LIST OF FIGURES

Figure 1: LTE architecture.....	2
Figure 2: Standard EPS-AKA.....	5
Figure 3: Overview of proposed REPS-AKA protocols.....	10
Figure 4: Four backends of AVISPA tools.....	13
Figure 5: General overview of Scyther tools.....	16
Figure 6: Steps of REPS-AKA1 protocol.....	26
Figure 7: Evaluation result in OFMC.....	32
Figure 8: Evaluation result in CL-AtSe.....	32
Figure 9: Scyther evaluation for defined claims in the UE (REPS-AKA1).....	33
Figure 10: Scyther evaluation for defined claims in the MME (REPS-AKA1).....	34
Figure 11: Scyther evaluation for defined claims in the HSS (REPS-AKA1).....	34
Figure 12: Evaluation of bandwidth consumption in core traffic.....	45
Figure 13: Bandwidth consumption of core traffic (when $n=7$).....	45
Figure 14: Computational cost in AKA protocols.....	47
Figure 15: Computational cost in AKA protocols (when $n=7$).....	47
Figure 16: Storage overhead in AKA protocols (when $n=5$).....	48
Figure 17: Steps of REPS-AKA2 (Ph1).....	51
Figure 18: Steps of REPS-AKA2 (Ph2).....	55
Figure 19: Evaluation result in OFMC.....	58
Figure 20: Evaluation result in CL-AtSe.....	58
Figure 21: Scyther evaluation for defined claims in the UE.....	60
Figure 22: Scyther evaluation for defined claims in the eNodeB2.....	60
Figure 23: Scyther evaluation for defined claims in the MME.....	61

Figure 24:Scyther evaluation for defined claims in the eNodeB1	61
Figure 25: Comparison of communication overheads	73
Figure 26: Comparison of <i>TP</i> in AKA methods	74
Figure 27: Fake and legitimate eNodeBs.....	76
Figure 28: Keys generation model of REPS-AKA3 Protocol	79
Figure 29: Steps of REPS-AKA3 (EPS security context)	79
Figure 30: Steps of REPS-AKA3 (NAS/AS Security Context)	83
Figure 31: Steps of REPS-AKA3 (Re-authentication)	86
Figure 32: Evaluation result in OFMC	89
Figure 33: Evaluation result in CL-Atse.....	89
Figure 34: Scyther evaluation for defined claims in the HSS (REPS-AKA3)	91
Figure 35: Scyther evaluation for defined claims in the UE (REPS-AKA3)	91
Figure 36: Scyther evaluation for defined claims in the MME (REPS-AKA3)	92
Figure 37: Scyther evaluation for defined claims in the eNB (REPS-AKA3)	92
Figure 38: Bandwidth consumption (Core Traffic)	107
Figure 39: Bandwidth consumption of core traffic (AV=5)	108
Figure 40: Total Bandwidth consumption	108
Figure 41: Total bandwidth in AKA protocols (AV=20)	109
Figure 42: Total bandwidth consumption in AKA protocols	110
Figure 43: Computational cost analysis of AKA protocols.....	112
Figure 44: Computational cost in AKA protocols (AV=20)	112
Figure 45: Storage overhead in AKA protocols (AV=5).....	114
Figure 46: LWI architecture	117
Figure 47: UE movements in LWI system	118
Figure 48: General diagram of UE selections for AKA protocols	119

Figure 49: Procedure of AKA selection	119
Figure 50: Steps of REPS-AKA4 Ph1.1	120
Figure 51: Steps of REPS-AKA4 Ph1.2	123
Figure 52: Steps of REPS-AKA4 Ph2.1	125
Figure 53: Steps of REPS-AKA4 Ph2.2.1	128
Figure 54: Steps of REPS-AKA4 Ph2.2.2	131
Figure 55: Steps of REPS-AKA4 Ph3.1	133
Figure 56: Steps of REPS-AKA4 Ph3.2	135
Figure 57: Steps of REPS-AKA4 Ph4.1	137
Figure 58: Steps of REPS-AKA4 Ph4.2	140
Figure 59: Evaluation results of REPS-AKA4 in OFMC mode	142
Figure 60: Evaluation results of REPS-AKA4 in CL-AtSe mode	142
Figure 61: Scyther evaluation for UE (REPS-AKA4 Ph1.1)	151
Figure 62: Scyther evaluation for MME (REPS-AKA4 Ph1.1)	151
Figure 63: Scyther evaluation for HSS (REPS-AKA4 Ph1.1)	152
Figure 64: Bandwidth consumption of core traffic	167
Figure 65: BW_{mh} of the AKA protocols ($AV=20$)	167
Figure 66: Bandwidth consumption of total traffic	168
Figure 67: Bandwidth consumption of total traffic ($AV=20$)	169
Figure 68: Computational cost of AKA protocols	170
Figure 69: Computational cost in AKA methods (when $AV=20$)	171
Figure 70: Evaluated scenarios (i.e., 1-6)	172
Figure 71: Comparison of average authentication delay	176
Figure 72: Comparison of signaling cost	178
Figure 73: Comparison of average signaling cost	179

Figure 74: Architecture of LTE-enabled IoT system.....	183
Figure 75: Steps of REPS-AKA5 (First Authentication phase)	187
Figure 76: Steps of REPS-AKA5 (Complete re-authentication)	192
Figure 77: Steps of REPS-AKA5 (External re-authentication)	194
Figure 78: Steps of REPS-AKA5 (Internal re-authentication)	196
Figure 79: Steps of REPS-AKA5 (Group member updating)	200
Figure 80: Evaluation result in OFMC	202
Figure 81: Evaluation result in CL-AtSe	202
Figure 82: Scyther evaluation for defined claims in the IoT-D (REPS-AKA5)	203
Figure 83: Scyther evaluation for defined claims in the MME (REPS-AKA5) ...	204
Figure 84: Scyther evaluation for defined claims in the IoT-L (REPS-AKA5).....	204
Figure 85: Scyther evaluation for defined claims in the HSS (REPS-AKA5)	205
Figure 86: The evaluation results of <i>TBC</i> for group-based AKA protocols	224
Figure 87: The evaluation results of BW_{mh} of AKA protocols	227
Figure 88: The BW_{mh} of AKA protocols for different AVs	228
Figure 89: Computational cost in AKA protocols	230
Figure 90: The <i>CC</i> of AKA protocols for different AVs.....	231
Figure 91: The <i>CC</i> of Group-based AKA protocols for different AVs	233
Figure 92: The signaling overhead of AKA protocols for different “ <i>n</i> ” and “ <i>m</i> ” .	234
Figure 93: Storage overhead in AKA protocols (AV=5).....	235
Figure 95: Summary of the REPS-AKA protocols.....	242

LIST OF SYMBOLS AND ABBREVIATIONS

(A) Sym	(A) Symmetric
(N) AUTH _m	(New) Auth. Token of m th Group Leader
(N) KASME _m	(New) KASME of m th Group Leader
(N) UHNI	(New) Updated Home Network ID
(N) XRES _{Lm}	(New) XRES of m th Group Leader
(N/T) AUTH _{n-m}	(New/Temporary) Auth. Token of IoTD _{n-m}
(N/T) SKDL _{n-m}	(New/Temporary) Secret Key Between IoTD _{n-m} and IoTL _m
(U) KASME	(Updated) Key Access Security Management Entries
(U) RAND	(Updated) Random Number
(U) RES	(Updated) Response
(U) XRES	(Updated) Expected Response
(X) e1ueAuth*	(Expected) New e1ueAuth
(X) e2ueAuth*	(Expected) e2ueAuth
AAA	Authentication Authorization Accounting
AAT	AKA Advertising Table
AD	Authentication Delay
AK	Anonymity Key
AKA	Authentication and Key Agreement
AMF	Authentication Management Field
AMS	Average Message Size
AP	Access Point
API	Application Program Interface
APID	Access Point ID

AS	Access Stratum
AS-ALG-C	AS Algorithm Category
AS-Alg-ID	AS Algorithm ID
AS-EAD	AS Encryption Algorithm Distinguisher
AS-IAD	AS Integrity Algorithm Distinguisher
AS-UPEAD	AS User Plane Encryption Algorithm Distinguisher
AT	Authentication Type
AuthFunc	Authentication Function
AUTN	Authentication Token
BMS	Building Management Systems
BW	Bandwidth Consumption
CC	Computational Cost
CK	Cipher Key
CL-AtSe	CL-based Model-Checker
CMID	Connected MME ID
CT	Core Traffic
DB	Download Backhaul
DCID _{n-m} /DTID _{n-m}	Device Constant/Temporary ID
DDoS	Distributed Denial-of-Service
DR	Download Radio
e1ueAuth	Auth. Parameter Between UE and eNB1
e2ueAuth**	New e2ueAuth* for Next Handover
EAP-AKA	Extensible Authentication Protocol-AKA
ECDH	Diffie–Hellman
eID1, eID2	Identifier of eNB1 and eNB2

eNodeB	evolved NodeB
EPC	Evolved Packet Core
FDT	Final Decision Table
FeID	Identity of Fake eNodeBs
FMID	Fake MME Identity
GAV	Group Authentication Vector
GID _m	ID of m th Group
GIK _m / GXK _m	Group Internal/External Key
GLL	Group Leader List
GL _m /GLL _m	List of IoTDS in m th Group/Group Leader List
H2C	Human to Computer
H2H	Human to Human
HC	Hop Count
HHVP	Hash of HVP
HI	HSS Identifier
HID	HSS Identity
HIMSI	Hash of IMSI
HLPSL	High-Level Protocol Specification Language
HMVP	Hash of MVP
HNID	Home Network Identity
HRES	Hash of RES
HRu	Hash of Ru
HSNID	Hash of SNID
HSS	Home Subscriber Server
HVP	HSS Validation Parameter

IF	Intermediate Format
IK	Integrity Key
IMSI	International Mobile Subscriber Identity
IoTDevice	IoT Device
IoTLeader	IoT Leader
IoTS	IoT Servers
KASME*	New KASME
KDF	Key Derivation Functions
KeNB	Session Key
KGC	Key Generate Center
Kmh	Key Between HSS and MME
KNASenc	Key of NAS Encryption
KNASenc*	Updated KNASenc
KNASint	Key of NAS Integrity
KNASint*	Updated KNASint
Kph	Public Key of HSS
Kpm	Public Key of MME
Kprm	Private Key of MME
Kpu	Public key of UE
KRRCenc	Key RRC Cipher
KRRCenc*	Updated KRRCenc
KRRCint	Key of RRC Integrity (AS Layer)
KRRCint*	Updated KRRCint
KRRCUPenc*	Updated KRRCUPenc
KRRUPenc	Key RRC User Plane cipher

Kum	Key Between UE and MME
LAAA	LTE AAA
LAPID	Last Access Point ID
LAUTHP	Last Authentication Parameter
LNI	Last Network ID
LTE	Long Term Evolution
LUI	LTE User Identity
LUIFunc	Last UI Function
LWI	LTE-WLAN Interworking
M2M	Machine to Machine
MA	Multiple Authentication
MAC	Message Authentication Code
MD	Mean Density
MK/UPK	Master Key/Updated key
MME	Mobility Management Entity
MSK	Master Session Key
MTC	Machine Type Communication
MTCD	MTC Device
MVP	MME Validation Parameter
MXRES	Master XRES (Expected Response)
NAI	Network Access Identifier
NAS	Non-Access Stratum
NAS-ALG-C	NAS Algorithm Category
NAS-Alg-ID	NAS Algorithm ID
NAS-CAD	NAS Ciphering Algorithm Distinguisher

NAS-IAD	NAS Integrity Algorithm Distinguisher
NAS-UP-C	NAS Uplink Count
NAUTN	New AUTN
NDI _{n-m}	New Device ID of n th IoT-D in m th Group
NeID	Neighbor eNodeB's ID
NIFunc	Network Identity Function
NIGFunc	NI Generation Function
NK	New Key
NKASME	New KASME
NKeNB	New KeNB
NKFunc	NKG Function
NKG	New Key Generator
NKGFunc	New Key Function
NLNI	New LNI
NMK	New MK
NNI	New Network Identity
NNIGFunc	New NI Generation Function
NNK	New and Novel Key
NRES	New RES
NTM	Number of Transmitted Messages
NUI	New User Identity
NUIFunc	NUI Function
NXRES	New XRES
OFMC	On-the-fly Model-Checker
OSK	Original Symmetric Key

PBS	Perfect Backward Secrecy
PFS	Perfect Forward Secrecy
P-GW	Packet Data Network Gateway
PKap	Public Key of AP
PKap	Public Key of AP
PKenb	Public Key of eNodeB
PKFenb	Public Key of fake MME
PKFmme	Public Key of fake MME
PKls	Public Key of LAAA
PK_{n-m}/PK_{glm}	Public Key of $IoT D_{n-m}/IoT L_m$
PK_{mme}/PK_{hss}	Public Key of MME/HSS
PKue	Public Key of UE
PLUI	Pervious LUI
R	Rate of Authentication Request
RES	Response
RK	Random Key
RNap1/2	Random Number of AP1/2
R_u, R_{Nue}	Random Numbers in UE
RNls	Random Number of LAAA
RN_{n-m}/RN_{gm}	Random Number of $IoT D_{n-m}/IoT L_m$
RNws	Random Number of WAAA
RRC	Radio Resource Control
SA	Successful Attach
SC	Signaling Cost
SCS	Service Capability Server

S-GW	Serving Gateways
SIG _m	Secret Identity of m th Group
SKap2ws	Shared Key Between AP2 and WAAA
SKap3ws	Shared Key Between AP3 and WAAA
SKapws	Shared Key Between AP and WAAA
SKem	Secret Key Between the eNodeB and MME
SKmh	Secret Key Between the MME and the HSS.
SK ^{n-m}	Secret Key of Between n th IoT-D in m th Group
SKwsls	Shared Key Between LAAA and WAAA
SL	Signal Level
SNID	Service Network Identity
SO	Signaling Overhead
SQN	Sequence Number
ST	Suspicious Table
StO	Storage Overhead
TBC	Total Bandwidth Consumption
TDI _{n-m}	Temporary Device ID of n th IoT-D in m th Group
T _m	Timestamp Generated by m th Group Leader
TS	Timestamp
UAUTN	Updated AUTN
UB	Upload Backhaul
UE	User Equipment
UECap	UE Capability for Encryption
UI	User Identifier
UID	User ID

UIGFunc	UI Generation Function
UIH	UE Identifier for HSS
UK	Updated Key
UKFunc	UK Function
UKG	Updated Key Generator
UKGFunc	UK Generator Function
UNID	Updated Network ID
UNIDFunc	UNID Function
UPKFunc	Update Key Function
UR	Upload Radio
WAAA	WLAN AAA
WLAN	Wireless LAN
WLID	WLAN ID
XHHVP	Expected HHVP
XHIMSI	Expected HIMSI
XHMVP	Expected HMVP
XHNID	Expected HNID
XMA	Expected MA
XMAC	Expected MAC
XRES	Expected Responses
XRES _{n-m}	XRES of n th IoT-D in m th Group
XRNK	Expected RNK
XUIH	Expected UIH

Chapter 1

INTRODUCTION

Nowadays, Internet of Things (IoT) technologies become popular considerably. The IoT systems connect several smart devices to provide the required services. The IoT applications employ Machine to Machine (M2M) or Machine Type Communications communication (MTCs) to meet the requirements of IoT applications. To develop IoT systems, one of the most important challenges is providing a proper security level for each IoT Device (IoTD). The IoT systems are effective and popular when all the IoTD are verified successfully. The LTE network is an appropriate network for IoT systems. One of the most important objectives of an LTE-enabled IoT network is to manage a great number of IoTDs with a proper level of security. When plenty of IoTDs connects to the LTE network simultaneously, all the IoTDs need to run a full AKA process, which might make problematic signaling congestion. Thus, supporting a great number of IoTDs is a fundamental requirement of IoT systems.

The introduction chapter is reviewed in four sections. In section 1.1, the architecture of the LTE network has been mentioned. We focus on needed infrastructure (i.e., nodes) which are related to security mechanisms in LTE networks. In section 1.2, the standard AKA method of LTE network is presented. Then, in section 1.3, we present security loopholes of LTE system. In addition, the main contributions of our academic thesis are given in section 1.4.

1.1 Architecture of LTE network

We explain architecture of LTE network as shown in Figure 1 [1-5]. The LTE system includes Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC) [1-25].

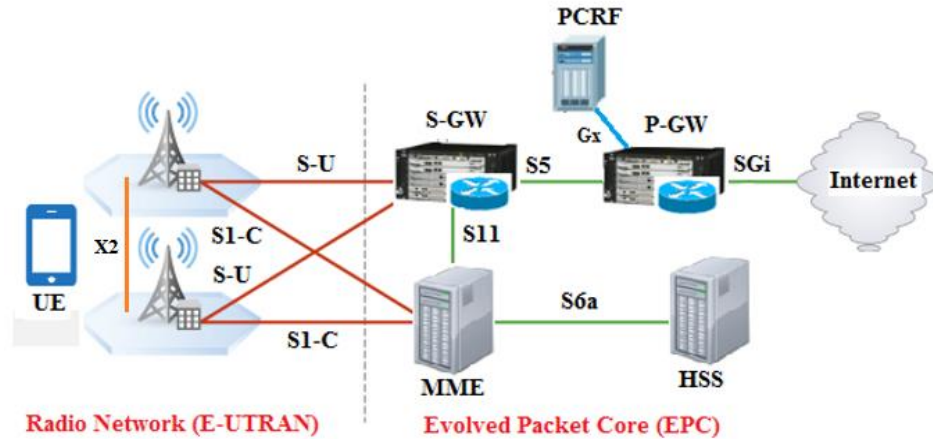


Figure 1: LTE architecture

In the E-UTRAN, User Equipment (UEs) are connected to evolved Node B (eNodeB), while the EPC section contains different LTE nodes to connect UEs to the Internet. The main components of the LTE system are listed as follows [4-15]:

eNodeB: The eNodeBs are the access points of an LTE network that handle the wireless communications in the network. The eNodeBs are directly connected to EPC (i.e., LTE core network) using the S1 interface. The eNodeBs are responsible to control/manage the wireless radio channels of LTE network. Each eNodeB can be interconnected with another one through the X2 interface, especially during handover procedures. We can categorize LTE access points into two main types including common eNodeB and Home Evolved Node B (HeNB). The HeNB works as a low-power eNodeB like a WiFi access point in wireless LAN in a small office. It can improve indoor LTE coverage and the speed of data uploading/downloading. Wide Area Networks (WANs) such as the Internet need to be used to set connections

between each HeNB and other common eNodeBs. Each eNodeB is an active entity of the security architecture of the LTE network, and it plays an important role in AS Security Context (in LTE-AKA procedure). In the AS security context, main session keys are generated to provide integrity and confidentiality of transmitted data.

PDN Gateway (P-GW): The P-GW is a bridge between the LTE network and external IP-based networks to transfer the UE's traffic. It is responsible for some services such as IP allocation/reallocation, and data filtering. It makes a connection between UEs and external Public Data Networks (PDNs) when it works as an exit/enter point for in/output traffic. Moreover, to access several PDNs, LTE mobile users (i.e., UEs) might need to use different P-GWs.

Serving Gateway (S-GW): This node is responsible for routing and forwarding the UE's packets. The S-GWs control the mobility of UEs between different eNodeBs in the LTE system. It is a connection between eNodeBs in E-UTRAN and P-GW in the LTE network to manage UE's mobility. Moreover, the S-GW can help inter-eNodeB handovers in order to make mobility services between LTE users and other communication networks (e.g., 2G and 3G systems).

Mobility Management Entity (MME): This is the most important LTE node that controls security functionalities. Importantly, this node performs the challenge-response procedure of the EPS-AKA method. The MME node is placed in the serving network named visited network. It is responsible for Access Security Management Entity (ASME) to handle LTE security context. Additionally, Non-Access Stratum (NAS) security context covers the security of communications between MME and the UE. When each UE makes a connection to the LTE core network (i.e., EPC), the MME

node very firstly requests the needed authentication information from the HSS in home network of subscribers. The MME needs the security information of each UE to run the AKA procedure with the attached UE.

Home Subscriber Server (HSS): This node is a central database containing UE's AKA parameters. The HSS forwards EPS Authentication Vector (EPS-AV) towards the MME in the EPS-AKA protocol. This node stores not only the security information of the LTE users but also generates the Authentication Vectors (AVs) and main keys which are used in AKA process. In addition, it stores LTE device information used for device triggering and authorization purposes.

Policy and Charging Rules Function (PCRF): This entity supports the service policies of LTE systems. It also manages and controls data sessions. All the quality of service (QoS) rules are distributed to P-GW by the PCRF which provides control/management of LTE services. In addition, the PCRF helps to provide an opportunity for other mobile users in other networks to connect to LTE users.

1.2 Security architecture of LTE system

The 3rd Generation Partnership Project (3GPP) defined the EPS-AKA as a standard security protocol for LTE system. It performs challenge-response mechanisms between the UE on behalf of user and MME on behalf of the network as shown in Figure 2 [5-12]. The steps of the EPS-AKA method are illustrated as follows [12] [20-45]:

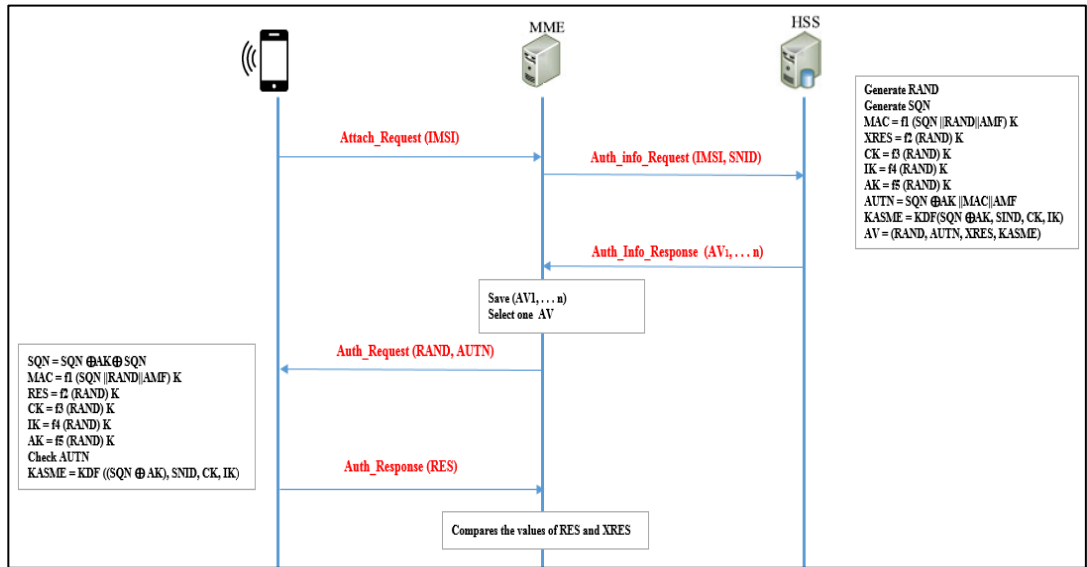


Figure 2: Standard EPS-AKA

1. UE → MME (Attach_Request message): The UE receives user identity request as beacon message from eNodeB and forwards its International Mobile Subscriber Identity (IMSI) to network side (i.e., MME).

2. MME → HSS (Auth_Info_Request message): The MME receives Attach_Request message from UE and checks the received IMSI. Then, MME attaches its Service Network Identity (SNID) to Auth_Info_Request message and forwards it to the home network (i.e, HSS).

3. HSS → MME (Auth_Info_Response message): The HSS gets Auth_Info_Request message and checks the IMSI and the SNID values. Then, it generates an Authentication Token (AUTN), a master key for encryption (KASME), a Random number (RAND), and an expected response (XRES) for each authentication vector (i.e., EPS-AV). The HSS sends them to the MME. The HSS employs the secret key of user (i.e., K), and two types of security functions to make AVs. The f1, and f2 are authentication functions; and the f3, f4, and f5 are key generation functions.

4. MME → UE (Auth_Request message): When the MME gets Auth_Info_Response message from HSS, it forwards Auth_Request message including a RAND and an AUTN to each UE. The MME selects another AV (i.e., unused parameters) for next AKA procedure.

5. UE → MME (Auth_Response message): When the UE gets Auth_Request message from the MME, it calculates the value of the expected Message Authentication Code (XMAC). Then, the UE compares the calculated XMAC with the received one in AUTN to verify network side. After a successful verification, the UE produces the Response (RES) and forwards this value to the MME using Auth_Response message. Then UE generates the master key (i.e., KASME).

6. MME: The MME compares the received RES with saved XRES to confirm this match, then it sends a success message to the target UE. This message means that AKA procedure is done successfully.

The general procedure of the proposed AKA protocols (i.e., REPS-AKA methods) is the same as EPS-AKA protocol. It means that there is no need for additional nodes in the proposed protocols. Thus, all REPS-AKA methods can be adapted to existing LTE infrastructure. In addition, signaling overhead (i.e., number of transmitted messages), and receiver/transmitter nodes for each step of our presented methods are the same as EPS-AKA protocol. In the proposed methods, we generate needed keys and parameters by the embedded key which is set in manufacturing time. We have the same approach in EPS-AKA method. In addition, we employ different random numbers to generate our security parameters. In EPS-AKA method, we also use a

random number. The procedure of key and parameter generation are different in our proposed methods and EPS-AKA protocol.

1.3 Security loopholes of LTE network

We present the security vulnerabilities of the EPS-AKA as follows [12] [30-65]:

IMSI cracking: The user's identification is the first step in AKA protocols. The identification step in EPS-AKA is performed by transmitting the user's IMSI in plaintext form over the LTE channel. Thus, attackers can extract the IMSI parameter in an specified area. Capturing a user's identifier (i.e., IMSI) allows the attackers to detect the UE's location. In addition, the adversary might extract the user's temporary identifier which sent over an unsecured air interface in the LTE network. When the victim (LTE user) tries to connect to a fake eNodeB (i.e., attacker's node), this eNodeB can force the UE to transmit its main identity (i.e., IMSI). Thus, the adversary recovers transmitted information and can trace the LTE users.

Lack of mutual authentication: In EPS-AKA, the authentication is indirect. The UE uses AUTN to authenticate the HSS, and the MME checks RES to confirm the UE. However, the UE cannot verify the MME on one hand, and the MME cannot verify the HSS on the other hand.

Man in the Middle (MitM) attack: An adversary can send a fake Attach_Request message using a valid UE's identity (from a captured user) to the MME. At that time, the MME forwards authentication parameters (i.e., RAND and AUTN) back to the intruder. Then, the attacker waits for an Attach_Request message from that genuine UE. The UE requests a new attachment, and the attacker sends previously received authentication parameters to the victim UE. Therefore, the attacker authenticates itself

to the victim UE as a genuine network node. By running the MitM attacks, an attacker can make an unsafe connection between the two LTE entities and start to modify/intercept/remove transmitted messages in data and control planes. The MitM attacks in LTE systems are feasible because authentication procedures among the communication nodes are handled in one direction.

Unprotected Wireless Interface: The wireless interface in the LTE network is always at risk of security threats such as manipulation and eavesdropping on user's packets. For example, the UE forwards an Attach_Request message to the LTE nodes based on EPS-AKA method. The Attach_Request message is transmitted over an unsafe wireless channels. If a malicious attacker eavesdrops on these channels, s/he can perform serious attacks such as MitM attacks.

SNID leakage: In the Auth_Info_Request message, the MME transmits its SNID to the HSS in plaintext form through the LTE channels. An attacker can capture the Auth_Info_Request message to detect the identity of the involved MME.

DoS attacks: The Denial of Service (DoS) attack prevents legitimate UEs from using the requested services in the LTE network. The LTE system allows unprotected transmissions during the AKA procedure. Thus, a malicious attacker can launch various active attacks against the unprotected AKA messages. Consequently, the UE can not obtain its requested services.

Replay attack: Replay attacks were in previous generations of networks (e.g., GSM, and UMTS) and are still possible in current LTE networks. In the LTE system, an attacker can capture the authentication parameters (i.e., RAND and AUTN)

transmitted by the MME and replay them to the victim UE. Thus, an attacker can find the presence of the victim UE by checking the received message. Hence, the current AKA method is vulnerable to this security attack.

Redirection attack: In the EPS-AKA method, there is not an strong mutual authentication among the LTE nodes. Thus, an adversary can place a fake eNodeB that will be seen as a legitimate one in the network. Then, the rouge eNodeB (i.e., attacker) sends the AKA messages to the victim UEs. Thus, this rouge eNodeB can redirect the UE's request message to the fake LTE system.

Bandwidth consumption and overhead storage issues: The bandwidth consumption between HSS and MME can be very high. This issue is a performance-decreasing factor in EPS-AKA protocol. In addition, when the UE resides in the coverage of of MME node for a long time, it may use all its saved EPS-AVs. Then, the MME requests another set of EPS-AVs from the HSS again. This generates more bandwidth consumption among LTE nodes. This issue also makes more storage overhead at the MME's database because it needs to save more EPS-AVs.

Single-key dependency: In the EPS-AKA scheme, the security of the LTE networks is based on a single symmetric key (i.e., K) pre-shared between the UE and the HSS. Therefore, if attackers extract this key, all dependent keys can be extracted.

Issue of KFS and KBS: The Key Forward/Backward Secrecy (KFS/KBS) is an important issue in AKA methods. In the EPS-AKA protocol, the eNodeB generates the new keys for AS layer from the current key (i.e., K_{eNB}). When the eNodeB is

captured, the dependent keys can be retrieved. Thus, the KFS/KBS feature can not be satisfied in the standard EPS-AKA protocol properly.

Tracking temporary identity of LTE users: In the LTE system, each UE has a Globally Unique Temporary ID (GUTI). These GUTI values are assigned to the LTE devices (i.e. UEs) by the MME node. If the GUTI values are not updated regularly, an attacker can monitor them. And if the attacker finds GUTI, s/he can use it to trace targeted UE. Then, by monitoring the LTE's user, when it connects to the attacker's server, the attacker can extract important information from users. This occurs when LTE's user logs into a fake website controlled by that attacker.

1.4 Thesis contribution

The key contributions of our thesis are listed as follows:

1. In this research, we evaluate and categorize the most important security issues in LTE network and LTE-enabled IoT systems. Taking into consideration the important performance and security issues, we present five security protocols for LTE systems and LTE-enabled IoT applications as shown in Figure 3.

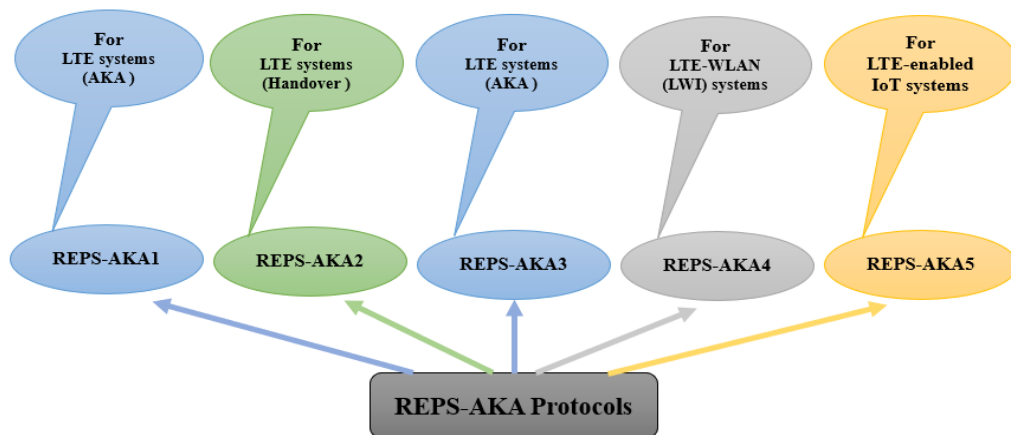


Figure 3: Overview of proposed REPS-AKA protocols

2. The presented methods provide strong mutual authentication and proper key management between user side (i.e., UE or IoT device) and the network side. The suggested methods generates a set of security parameters in each involved entities to satisfy the node's privacy during the authentication procedure.
3. The proposed AKA protocols follow the characteristics of the 3GPP standards (i.e., current LTE system) by using Symmetric/Asymmetric key infrastructure and calculated security parameters. These security approaches solve single key issue by generating the secret keys based on random values and some security parameters.
4. The presented protocols provide the unlink ability in the LTE networks and LTE-enabled IoT systems. Thus, they satisfy the KFS/KBS for involved nodes. By proposed security methods, a secure session key can be generated between UE and eNodeB to establish a safe communication in the LTE system.
5. The suggested methods considerably improve performance metrics such as bandwidth consumption, computational cost, signaling overhead, and storage overhead. Thus, they can provides a suitable QoS level in the systems.
6. Formal security analyses of the presented methods are accomplished by using BAN logic, Scyther, and AVISPA tools. The evaluation results prove that the presented methods satisfies all defined security goals and the protocols are resistant to known security attacks.

Chapter 2

LITERATURE REVIEW

The literature review of this thesis is presented in three sections. A general description of used security verification tools is provided in section 2.1. A set of important related works in LTE security methods is presented in section 2.2. Then, in section 2.3, we review several group-based security protocols in LTE networks which can be presented for LTE-enabled IoT systems.

2.1 Simulations and security tools

Designing security protocols is one of the most important subjects in both industrial and academic environments. Each protocol needs to work under worst-case scenarios in open networks such as the Internet. Accordingly, verification tools to support analysis of security protocols are important point in improving and development of security protocols. There are several appropriate verification tools to check AKA protocols based on defined security goals. In this thesis, we selected AVISPA, Scyther, and BAN Logic as our simulation/verification platforms. Thus, we provide a general description of these security tools/logic in this section [12-16] [32-49].

2.1.1 AVISPA verification tool

The AVISPA is a strong pushbutton tool for verification of security methods. It employs a modular language to modeling security methods and the security goals. By using this tool, we can verify a large set of network security protocols, this prove that the AVISPA tool is the state of the art for verification of different AKA protocols.

The AKA protocols need to be written in HPSL which is High-Level Protocol Specification Language. The HPSL is a modular language to present intruder models and security properties. These points make HPSL an appropriate option for specifying academic and industrial scale security protocols. The HPSL code must be translated into an Intermediate Format (IF). The IF is a lower-level language in abstraction viewpoint. The HPSL codes are converted into the IF by means of HPSL2IF translator as shown in Figure 4. The IF code is utilized as an input to the back-ends part of the AVISPA tool. There are four main back-ends modes in AVISPA including Tree-Automata-based Protocol Analyzer (TA4SP), SAT-based Model-Checker (SATMC), On-the-fly Model-Checker (OFMC), and CL-based Attack Searcher (CL-AtSe). These back-ends can verify that whether a presented method is safe or not according to the defined goals.

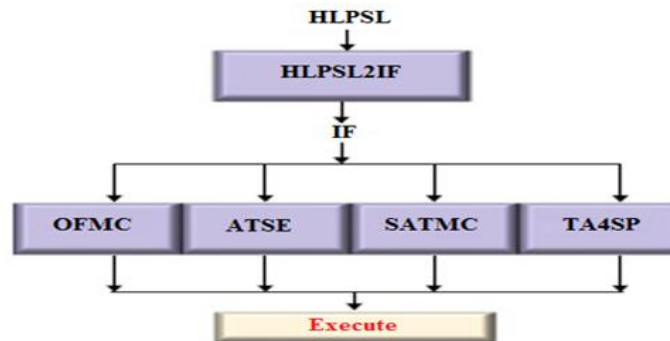


Figure 4: Four backends of AVISPA tools

We employ the AVISPA animator named Security Protocol Animator (SPAN) to write the HPSL codes of the protocol. The SPAN is designed to help developers in writing HPSL codes. It simulates the security attacks by modeling an active intruder. If a security attack is found on a modeled protocol, the SPAN displays it by a message sequence chart. The AVISPA uses Dolev-Yao attack model to check AKA protocols. In this model, we assume that the system is completely monitored by an attacker.

2.1.2 Scyther verification tool

The Scyther is another verification tool for formal security analysis. This security tool is used to find security issues that launch from the way the modeled method is implemented (based on defined cryptography assumption). The Scyther can provide useful evaluation results even in the situation that there is no attack. In these cases, the evaluation results have a output result like a bounded verification tool. In addition, this tool provides different classes of protocol behavior in particular for classes of attacks. This verification software can facilitate so-called multi-protocol analysis. Accordingly, this tool can be successfully used in academic and industrial environments. The proposed protocols are modeled in SPDL which is Security Protocol Description Language and it is based on the operational semantics. By using SPDL in Scyther tool, we can define different security properties which are implemented as claim events. For instance, we can define a specified claim to prove that a transmitted parameter is confidential (secrecy). Then, the Scyther can check this claim to verify modeled protocol.

Each verification tool covers one or more security model. The Scyther tool uses Dolev-Yao' adversary model to check AKA methods. In the Scyther tool, the cryptographic functions are assumed to be perfect. Accordingly, the adversary cannot extract anything from encrypted messages unless s/he learns the secret keys. Based on Dolev-Yao model, there is no need to formalize the abilities of an adversary in our modeled protocols [47]. In the Scyther tool, an attacker (presented by " A ") can check transmitted messages on the network communication. In this section, we mention how an attacker " A " extracts new transmitted data. We can assume that " N " is the attacker's knowledge, and " f " is a function which we use to make the relations among different

sets of “ N ”. In addition, “ k ” can be a secret key where “ k^{-1} ” is its reverse. Accordingly, “ k^{-1} ” equals to “ k ” in symmetric key infrastructure. The concatenation of terms x_i and x_j can be presented by (x_i, x_j) . Accordingly, there are some assumption in Scyther: $x \in N \Rightarrow N \vdash x$: if x is an item of N , A (i.e., N) knows x ; $N \vdash (x_1, x_2) \Rightarrow \{N \vdash x_1, N \vdash x_2\}$: if A (i.e., N) knows (x_1, x_2) , then A (i.e., N) knows both terms x_1 and x_2 ; $\{N \vdash x_1, N \vdash x_2\} \Rightarrow N \vdash (x_1, x_2)$: if A (i.e., N) learns both x_1 and x_2 , A (i.e., N) knows (x_1, x_2) ; $\bigwedge_{1 \leq i \leq s} \vdash x_i \Rightarrow N \vdash f(x_1, \dots, x_s)$: if A (i.e., N) knows all x_i ($1 \leq i \leq s$) and f is a general (i.e., public) function, then A (i.e., N) is able to generate the output of function (i.e., f) by using the input $x_1, \dots, x_s$; $\{N \vdash x, N \vdash k\} \Rightarrow N \vdash \{x\}_k$: if A (i.e., N) learns message (i.e., x) and valid key (i.e., k), A (i.e., N) is able to make encrypted message (i.e., $\{x\}_k$); $\{N \vdash \{x\}_k, N \vdash k^{-1}\} \Rightarrow N \vdash x$: if A (i.e., N) learn the encrypted message (i.e., $\{x\}_k$) and the key k^{-1} , A (i.e., N) is able to decrypt cipher text and extract corresponding plaintext (i.e., x). Thus, an attacker A which has above information is able to monitor transmitted messages.

In the Scyther, we need to define different claims to specify security requirements. The security claims which are used in our protocol verification are Alive, Nisynch, Niagree, and Secret. Alive claim is a kind of authentication. Based on Alive claim, a given communication party (R) has executed some specified events (e.g., $claim(R, Alive)$). The modeled protocol guarantees to a node X aliveness of another node Y if, whenever X (as initiator) completes a given run of an AKA protocol, seemingly with node Y (i.e., as responder), then Y has previously been running the protocol. Nisynch claim can be defined to prove that received messages of R are sent by a node X (as sender) and are received by another node Y (as receiver). The Nisynch is defined as $claim(R, Nisynch)$. In Secret claim which is defined as $claim(R, secret, Q)$, if a term Q is claimed to be secret, Q should be kept secret to the attacker. Based on Non-injective

agreement (Niagree) claim, if a node X completes a protocol execution seemingly with node Y , then X has run the protocol with X , and node X agrees with all other transmitted values. The Scyther simulation is a well-known security verification tool that offers a graphical analysis of modeled protocol. A general overview of Scyther procedure for protocol verification is indicated in Figure 5.

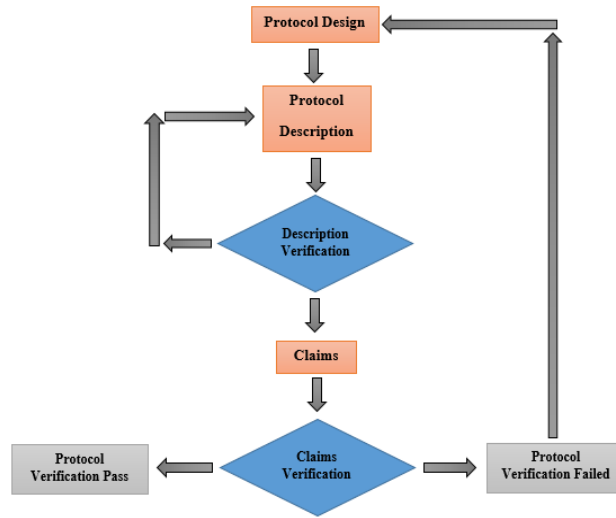


Figure 5: General overview of Scyther tools

2.1.3 BAN logic

The main purpose in formally analyzing a security method is to prove the correctness of the modeled method. The BAN logic is a well-known formal analysis for different AKA protocols. This logic is presented by Burrow, Abadi and Needham to model the central concepts of each AKA protocol. The BAN logic works with beliefs of trustworthy principals running AKA protocol. The evolution of defined beliefs are modeled as a consequence of transmitted messages between involved nodes. Accordingly, the BAN logic works appropriately in discovering security issues and redundancies in common used security methods. In the BAN logic, there are two times (present and past time) which are considered for each message. The present time

begins with the sending first message and ends with receiving last message. Each transmitted message which sent before this time can be considered as past time [48].

For the formal evaluation of an AKA protocol, the following assumptions are assumed including “ P ” and “ Q ” are the two nodes (i.e., principals); “ K_{pq} ” is the symmetric key; “ K_p ” and “ K_q ” show the public keys, “ K_p^{-1} ”, and “ K_q^{-1} ” are secret keys of “ P ” and “ Q ” respectively; “ X ” and “ Y ” work as two different statements. We provide a list of the basic notations of the BAN logic model as shown in Table 1.

Table 1: Notations of BAN Logic

Notation	Definition
$P \models X$	Node P can believe “ X ” : node P can be sure that “ X ” is true
$P \mid \sim X$	Node P can once say “ X ” : node P sends a message including “ X ”
$P \triangleleft X$	Node P can see “ X ” : node P can receive a message including “ X ”
$P \mid \Rightarrow X$	Node P has jurisdiction over “ X ” : node P can control “ X ”
$\#(X)$	“ X ” has not sent at any time before execution: “ X ” is fresh value
$\{X\}_K$	“ X ” encrypted with key K
$\xrightarrow{K_p} P$	Node P uses K_p as its public key (i.e., K_p^{-1} is its secret key)
$P \xleftrightarrow{K_{pq}} Q$	Nodes P and Q have shared secret key named K_{pq}
$P \xleftrightarrow{X} Q$	Only Nodes P and Q can know the secret formula “ X ”

Moreover, the combination of X and Y can be written as (X, Y) which is a formula in this Logic. The first rule of the BAN Logic is presented as follows: $\frac{X_1, \dots, X_n}{Y}$, Based on this rule, if all mentioned X (i.e., $X_1 \dots$ and X_n) hold, then we can assume that Y holds. Accordingly, the other different inference rules are listed as follows: **Message-meaning**: This rule works with the description of transmitted messages. We have three message-meaning rules. For shared keys: $\frac{P \models Q \xleftrightarrow{K} P, P \triangleleft \{X\}_K}{P \models Q \mid \sim X}$. Accordingly, if P believes that key K is shared between Q and P , and also P sees a message X encrypted by K , then P can believe that Q surely sent X . For public keys: $\frac{P \mid \Rightarrow Q, P \triangleleft \{X\}_{K^{-1}}}{P \models Q \mid \sim X}$. Accordingly, if P

believes K as the public key of Q , and sees a message X encrypted by K^{-1} (i.e., private key), then P believes that Q definitely sent X . For shared secrets:

$\frac{P| \equiv Q \stackrel{Y}{\leftrightarrow} P, P \triangleleft \{X\}_Y}{P| \equiv Q | \sim X}$. Accordingly, if P believes that secret Y is shared between P with Q , and

sees $\{X\}_Y$, then P can believe that Q transmitted X definitely; **Nonce-verification:** The

sender of a fresh message is supposed to believe it: $\frac{P| \equiv \#(X), P| \equiv Q | \sim X}{P| \equiv Q | \equiv X}$. Accordingly, if node

P believes that node Q transmitted value X in the recent run, then, node P can believe

that node Q believes X ; **Jurisdiction:** A node can trust a node on matters delegated to

the node: $\frac{P| \equiv Q | \Rightarrow X, P| \equiv Q | \equiv X}{P| \equiv X}$. Accordingly, if node P believes that node Q has jurisdiction

over message X and believes that node Q believes X , then node P can believe message

X ; **Belief rules:** The belief operators are characterize as follows:

$\frac{P| \equiv X, P| \equiv Y}{P| \equiv (X,Y)}, \frac{P| \equiv (X,Y)}{P| \equiv X}, \frac{P| \equiv Q | \equiv (X,Y)}{P| \equiv Q | \equiv X}$; **Utterance rule:** The "once said" operator is characterized

as: $\frac{P| \equiv Q | \sim (X,Y)}{P| \equiv Q | \sim X}$; **Sight rules:** The property of the "sees" operators is characterized as:

$\frac{P \triangleleft (X,Y)}{P \triangleleft X}, \frac{P \triangleleft \{X\}_Y}{P \triangleleft X}$; **Message decryption rules:** If node P receives a message encrypted by

key K , and K is shared between P and Q , then node P sees the X message (decrypted

message): $\frac{P| \equiv Q \stackrel{K}{\leftrightarrow} P, P \triangleleft \{X\}_K}{P \triangleleft X}$. If node P receives a message encrypted with its public key,

then P can see the decrypted message: $\frac{P| \equiv \rightarrow P, P \triangleleft \{X\}_K}{P \triangleleft X}$. If node P believes that K is the

public key of node Q , and node P sees a message encrypted by K^{-1} (i.e., private key)

then node P can sees the decrypted message X : $\frac{P| \equiv \rightarrow Q, P \triangleleft \{X\}_{K^{-1}}}{P \triangleleft X}$; **Freshness rule:** If P

believes that one (at least) component of a given formula is fresh, P can believes that

whole formula is fresh: $\frac{P| \equiv \#(X)}{P| \equiv \#(X,Y)}$; **Bidirectionality rules:** A shared key K or secret value

X can be used between a node P and Q in both directions as follows:

$$\frac{P| \equiv R \stackrel{K}{\leftrightarrow} R'}{P| \equiv R' \stackrel{K}{\leftrightarrow} R}, \frac{P| \equiv Q | \equiv R \stackrel{K}{\leftrightarrow} R'}{P| \equiv Q | \equiv R' \stackrel{K}{\leftrightarrow} R}, \frac{P| \equiv R \stackrel{X}{\leftrightarrow} R'}{P| \equiv R' \stackrel{X}{\leftrightarrow} R}, \frac{P| \equiv Q | \equiv R \stackrel{X}{\leftrightarrow} R'}{P| \equiv Q | \equiv R' \stackrel{X}{\leftrightarrow} R}.$$

2.2 Related works of LTE protocols

Many schemes and methods have been presented to address security vulnerabilities in LTE networks [45-93]. The suggested protocols are defined to improve the security level of the LTE system. Still, none of them has reached a perfect and practical solution so far. In [23], authors analyzed E-UTRAN protocols in the LTE system to show security vulnerabilities and practical attacks in particular DoS attacks. However, the approach required high bandwidth consumption over the LTE network. Abdrabou et al. [26] presented a security approach based on a symmetric key named MEPS-AKA. This protocol uses the concept of exponential key exchange. Ekene et al. [27] suggested a new AKA scheme based on public keys. In this protocol, some security parameters are sent in clear text, making this method vulnerable to some security attacks. Moreover, in the suggested method, the values of RES, RAND, and AUTN parameters are transmitted in clear text (i.e., unsafe), which leads to some security attacks. In [29], the authors presented a new method for LTE systems. The method solves some security weaknesses of the existing AKA protocol in the LTE system. However, the presented method requires high bandwidth among the LTE nodes. In addition, there are some security schemes for handover key agreements in LTE systems. In [30], authors proposed a new authentication method to make a secure the handover procedure between LTE system and Wireless LAN (WLAN). Han et al. [32] proposed a mathematical model to cover the de-synchronization attack. In this paper, authors show the security issues of handover key management in LTE system such as desynchronization attacks. The method can minimize the effect of desynchronization attacks but cannot be effectively prevented. Agarwal et al. [33] suggested a method to prevent de-synchronization attacks in LTE handover. However, this proposed method was not efficient since it cannot fully cover strong key agreement. The authors in [36]

proposed a secure handover scheme that requires a new attach method to verify the HSS node. In this method, they proposed a handover procedure for the LTE-Advanced (LTE-A) systems. This scheme proposed a registration process in order to enable the eNodeBs to verify the HSS node. Sridevi et al. [37] designed an approach for handover key agreement to increase the security levels between eNodeB and MME. In this paper, authors proposed a method which is device certification based scheme. The suggested method is implemented by using MATLAB software. The authors in [8] suggested an Efficient EPS-AKA protocol named EEPs-AKA to handle the security problems in LTE-AKA protocol, in which, the Password Exponential Key Exchange (SPEKE) method is used. Singh et al [11] explored different security issues like the weak handover issues by simulating the LTE system on NS3 software. They evaluated E-UTRAN protocols, and discovered some security threats in a real LTE environment. In [10], a security-enhanced AKA scheme named PSE-AKA protocol is proposed. This method is suggested to provide the privacy of LTE nodes. It could overcome some security issues in the LTE systems. However, it has high bandwidth consumption. Some researches proposed authentication methods for 3G-WLAN or LTE-WLAN systems [88-93] [97]. In [88], authors presented a security approach in WLAN and authors in [89] suggested authentication protocol for WiMAX networks. On the other hand, mostafa et al defined an AKA method in [84] to provide suitable security level in WLAN-LTE systems. Also, there are some other proposed security protocols for 3G-WLAN networks [90-93].

2.3 Related works of group-based protocols

In addition, some researchers presented several AKA protocols for M2M communications in LTE networks. These security schemes aimed to achieve security needs in IoT-enabled LTE systems. Garima et al [9] suggested a group based AKA

scheme for MTC named EMTC-AKA method. This method provides authentication between the MTC devices and integrity for wireless channels. The evaluation results show that suggested method can solve some security threats in the MTC network. Lai et al. [66] presented an E²SEC scheme for the MTC network and the LTE systems. The method is based on the proxy signatures to provide End-to-End security for MTC systems. The method employ EPS-AKA approach to establish authentication between the involved MTCDs. Thus, the scheme can be vulnerable to all the security threats which are in the EPS-AKA method. Modiri et al. [69] implemented an authentication protocol called GSL-AKA for M2M communications. The proposed scheme can overcome some security attacks of EPS-AKA protocols. Cao et al. [70] made a new method called UPPGHA protocol to verify a group of MTCDs. This method can achieve some security goals but it also suffers from communication overheads. Due to the unprotected wireless interface, the protocol can be threaten by some security attacks. Additionally, the method has high computational cost. Choi et al. [54] presented an AKA protocol to improve the grouping optimization in MTC systems. In this method, all the MTCDs and the LTE network (i.e., HSS) have a tree of binary values to make the MTC group. In this method, the addition of each MTCDs needs some computational cost. The protocol also uses unprotected interfaces that can be subjected to some security issues. Yao et al. [57] presented the group based method (GBS-AKA) to overcome some issues. However, this method fails to satisfy the privacy of involved nodes. Cao et al. [53] defined a new AKA protocol (group-based) named as GBAAM which uses identity-based cryptography approach. It needs to execute a full EPS-AKA procedure for each IoT node with some additional communication cost. Hence, the GBAAM protocol has all the security threats which are in the EPS-AKA method. Jiang et al. [52] designed a group based AKA method

named as EG-AKA using the elliptic curve Diffie–Hellman (ECDH). The EG-AKA method presented to protect the group identity of M2M system by setting some temporary IDs. However, the suggested method has a high computational cost, since it used ECDH algorithm. Lai et al. [59] proposed MTC-AKA protocol which is another group-based method for the MTC system. The method claims that it has a less computational cost. However, the protocol employs unprotected links that lead to some security issues such as MitM attacks. Lai et al [58], implement a security scheme to enhance authentication in particular for roaming services in mobile networks. But, it cannot satisfy important security issues such as integrity. A robust security protocol is presented in [12] for the AKA procedure in the LTE system. However, this method has a high computational cost for the LTE-enabled IoT system. Mou et al [1] defined an authentication protocol for LTE-enabled IoT applications. They claimed that the suggested method improves the performance metrics such as computational overhead. In this approach, they made two groups for each MTCD to reduce the complexity of the system. Parne et al. [68], [42] presented two protocols named as DGBES-AKA and SEGB-AKA to overcome signaling overhead. However, these methods generate inadequate authentication delay, especially during a handover procedure. Cao et al [49] presented an AKA method called as GAHAP to improve the existing handover procedure using a new group-based method. The method can handle a group of MTCDs in different mobility scenarios. The handover authentication is improved in this protocol but it is subject to signal congestion when there is a huge number of simultaneous MTCDs. To improve computational costs, Lai et al. [71] made a group based approach named as GLARM-AKA scheme. The method is suitable for resources of MTCDs but it is subject to impersonation attack.

Chapter 3

MODEL 1 (REPS-AKA1)

In this chapter, we presented the REPS-AKA1 protocol using a strong AKA mechanism with security improvements. The proposed protocol satisfies the main three security requirements for AKA. Namely, appropriate security level, converged AKA method, and proper QoS level. We also evaluate the REPS-AKA1 protocol according to security features by using BAN Logic, Scyther, and AVISPA tools. This chapter is organized as follows: In section 3.1, we present the REPS-AKA1 protocol in detail. Section 3.2 evaluates the suggested method by using verification tools. The performance evaluations of suggested protocol are provided in section 3.3.

3.1 The proposed REPS-AKA1 protocol

To avoid EPS-AKA's security issues in the LTE system, we proposed the REPS-AKA1. The main features of the suggested approach are presented as follows:

1. The REPS-AKA1 enables strong mutual authentication not only between UE and serving network (i.e., MME) but also between LTE entities such as HSS and MME.
2. The REPS-AKA1 method seriously reduces the signaling overhead and guarantees an acceptable level of QoS in the LTE networks.
3. The REPS-AKA1 is resistant to multiple security attacks including user anonymity, traceability, and DoS in the LTE networks.

4. Compared with standard EPS-AKA and some suggested AKA approaches, the REPS-AKA1 method has acceptable bandwidth consumption.

5. The proposed REPS-AKA1 follows the standard architecture of the LTE system and its security requirements defined in the 3GPP standards.

6. In the suggested method, we settle the problem of single key exposure using an updatable key shared between the UE and network side. This method provides acceptable secrecy and key freshness in the AKA process.

The EPS-AKA1 method has some security loopholes corresponding to the network secrecy and mobile users' privacy. The communication between UEs and LTE nodes on one side and between network nodes on the other side should be protected weaknesses. Therefore, we suggest the following modification to the original protocol:

1. The Symmetric and Asymmetric cryptography algorithms protect the communication interfaces between the used LTE nodes in the AKA process.

2. The REPS-AKA1 hides UE's identity (i.e., IMSI) from the third parties by hashing function (i.e., HIMS). Then, the HIMS is transmitted securely.

3. The UE uses New and Novel Key (NNK), which is a new calculated key. We generate NNK by performing the XOR (i.e., exclusive or) operator on generated Random Key (RK) and the Original Symmetric Key (OSK) shared between UE and HSS nodes. The main contribution of this new key (i.e., NNK) is to form a robust

secret key named as Kum between UE and MME. The Kum protects the channel between MME and UE in the authentication procedure.

4. The UE and the HSS employ a fresh security parameter, named UE Identifier for HSS (UIH), to authenticate each other. The main aim of the UIH is to protect the UE against impersonation attacks.

5. The MME and the HSS nodes in serving and home networks employ two new secret parameters in REPS-AKA1 method. These AKA's parameters are MME Validation Parameter (MVP) and HSS Validation Parameter (HVP). The HSS can authenticate MME by MVP. And The MME can authenticate HSS by HVP.

6. We use Hash of SNID (HSNID) in place of the original SNID to protect the network identifier. The REPS-AKA1 protocol transmits HSNID in the channel securely.

7. We utilize SNID and HMSI to generate Kum which is an asymmetric key shared among UE and MME nodes. These nodes can apply the Kum in future authentication procedures and decrease computational cost.

8. We added another unique parameter called Home Network Identity (HNID) which is calculated by MVP and HVP in the authentication procedure.

9. The proposed protocol has a practical feature of modifying Integrity Key (IK), Cipher Key (CK), and Anonymity Key (AK) in a given manner so that the keys are dynamic.

the UE executes the hash function (i.e., $H1$) on the IMSI parameter to generate HIMS_I: $HIMS_I = H1(IMS_I)$. Accordingly, the IMSI value can be sent to the MME securely. Then, UE produces a random value called R_u ; this random value will approve freshness of transmitted request message. Accordingly, the proposed method can protect MME node against Reply attack. We Hash the R_u : $HR_u = H2(R_u)$ to send it securely. The UE also has a list of MVP parameters to identify MME. MVP is distributed by eNB to all UEs in the eNB Zone. We hash it as well before transmission (Hash of MVP): $HMVP = H3(MVP)$. We generate UIH as an identifier between HSS and UE: $UIH = H4(IMS_I \oplus OSK)$. It is encrypted by HSS public key to increase the security level. Then, a new symmetric key (K_{um}) is generated based on IMSI and NNK that can help as a future symmetric key: $K_{um} = H5(IMS_I \oplus NNK)$. Finally, the Attach Request message includes two parts ($A1, A2$). The UE transmits $A1$ including HIMS_I, HR_u , $HMVP$, K_{um} , and $\{NNK\}_{inv}(K_{pu})$ encrypted by K_{pm} . The K_{pm} is public key of MME node in LTE system. And the UE forwards $A2$ including HIMS_I, $HMVP$, and UIH encrypted by public key of HSS (K_{ph}).

2. MME → HSS: The MME employs its private key (i.e., K_{prm}) to decrypt the received message and extract the $HMVP$, HIMS_I, HR_u , and K_{um} . The MME stores K_{um} for future transactions. It needs HIMS_I and HR_u to protect MME from DoS attacks. The serving MME looks for received HIMS_I in its database. It checks that if the UE is a new user or not. The MME can use HR_u to specify the number of attach requests from a given UE to this MME. If the MME finds unsafe behavior in a given interval, it will add that UE to a black list. The MME node hashes the SNID parameter (i.e., network identity) using the $H6$ function: $HSNID = H6(SNID)$. In this stage, a unique symmetric key (K_{mh}) will be calculated based on SNID and HIMS_I:

$K_{mh}=H7(SNID \oplus HIMS I)$. On the other hand, MME has the HVP of HSS, which is also hashed before transmission (Hash of HVP): $HHVP=H8(HVP)$. MME encrypts all of the parameters using the K_{ph} key and forwards Authentication Information Request (A2, A3). The A2 includes HIMS I, HMVP, and UIH encrypted by K_{ph} . And A3 contains K_{mh} , HHVP, HSNID, and $\{NNK\}_{inv}(K_{pu})$ encrypted by K_{ph} .

3. HSS $\rightarrow$ MME: The HSS decrypts the received message (i.e., A2 and A3) and extracts the values of K_{mh} , HSNID, HIMS I, HMVP, HHVP, and UIH. The HSS applies H1 to IMSI and generates Expected HIMS I (XHIMS I) to find UE in its database: $XHIMS I=H1(IMS I)$. To authenticate the MME, the HSS calculates Expected HMVP (XHMVP) and Expected HHVP (XHHVP): $XHMVP=H3(MVP)$, $XHHVP=H8(HVP)$, and compares them with the received ones (HMVP and HHVP). Then, HSS calculates Expected HNID (XHNID) based on MVP and HVP: $XHNID=H9(MVP \oplus HVP)$. After MME successful authentication, it employs the saved IMSI and OSK to generate the Expected UIH (XUIH): $XUIH=H1(IMS I \oplus OSK)$, and compares the generated parameter with the received UIH to confirm the UE. Then, the HSS calculates RNK (Received New Key) based on UIH: $RNK=H10(XUIH)$. Then, It produces Multiple Authentication parameters (MA): $MA=H11(RNK.HSNID)$. The MA value is a unique parameter between UE, MME, and HSS nodes. It will be used for the re-authentication procedure. Then, the HSS uses the RAND, Authentication Management Field (AMF), and Sequence Number (SQN) to calculate $MAC=F1(SQN||RAND||AMF)RNK$, $AK=F5(RAND)RNK$, $CK=F3(RAND)RNK$, $XRES=F2(RAND)RNK$, $AUTN=AUTHFunc((SQN \oplus AK)||AMF||MAC)$, $IK=F4(RAND)RNK$, and $KASME=KDF((SQN \oplus AK).HSNID.CK.IK)$. Finally, the HSS sends the calculated parameter to the MME using Authentication

Information Reply (A4, A5). The A4 includes HNID, SQN, RAND, AUTN, KASME, HRES, and XHMVP encrypted by K_{mh} . And A5 contains RNK, MA, and AMF encrypted by UE's public key (K_{pu}).

4. MME → UE: In this step, the MME node utilizes generated symmetric key (K_{mh}) to decrypt the Authentication Information Reply message. The MME verifies the HSS node by checking the HNID value (compare with the received HNID). After successful verification, the MME saves the received AVs and forwards needed parameters to the UE. The MME encrypts parameters by received K_{um} (in step 2) and sends them to UE. Consequently, the MME forwards the Authentication Request message (A6, A7) to the UE. The A6 includes SQN, RAND, AUTN, and HMVP encrypted by K_{um} . And the A7 contains RNK, MA, and AMF encrypted by K_{pu} .

5. UE → MME: In this step, the UE decrypts the Authentication Request messages using the generated symmetric key (i.e. K_{um}). The UE generates Expected RNK (XRNK) by hashing UIH: $XRNK = H_{10}(UIH)$ and compares XRNK with received RNK to authenticate HSS. Then, to authenticate the MME, the UE generates Expected MA (XMA) using received HSNID and XRNK: $XMA = H_{11}(XRNK, HSNID)$ and compares XMA with received MA. The UE generates HMVP to authenticate MME: $HMVP = H_3(MVP)$. After successfully authenticating the serving MME, the UE also calculates XAUTN and compares its value with the received AUTN to authenticate the HSS node again. Subsequently, UE will calculate XMAC, AK, RES, IK, KASME, CK, and XAUTN the same as HSS. After successfully authenticating the MME and the HSS, the UE makes Hash of RES (HRES): $HRES = H_{12}(RES)$ and encrypts it using generated K_{um} : $\{HRES\}_{K_{um}}$. Then, the UE transmits A8 including $\{HRES\}_{K_{um}}$ to the MME.

6. MME: The MME receives the HRES from the user's device (i.e., UE). Then, MME compares the received HRES with the saved XHRES to confirm the UE.

3.2 Security analysis of REPS-AKA1 protocol

The presented protocol is evaluated according to different security parameters in this section. We compare our suggested method with some other AKA protocols including EPS-AKA (standard method). The Security analyses of the REPS-AKA1 protocol are conducted using AVISPA tools, BAN Logic, and Scyther tool. These verification tools are the most well-known security tools to verify AKA protocols.

3.2.1 Formal analysis using AVISPA

The AVISPA tool is a security simulation to verify modeled protocols. In this section, we employ OFMC and CL-AtSe backends to check the validation of REPS-AKA1 protocol. The AVISPA tool can also show visual details about the occurred attacks during simulation. Otherwise, the final result displays that the simulated model is SAFE. The AVISPA tool employs a strong intruder model named Dolev-Yao. In this model, an intruder completely controls the network. All the transmitted messages between LTE nodes (i.e., UE, MME, and HSS) are reachable to the attackers. The verification's main goals are the mutual authentication and secrecy of transmitted parameters. The detailed security goals in REPS-AKA1 protocol are presented in Table 3 (AVISPA codes are presented in Appendix 1).

Table 3: Security goals in REPS-AKA1

Defined security goals	Description
<i>authentication_on_mme_ue_auth_HRES</i>	MME authenticated UE by HRES properly
<i>authentication_on_ue_hss_auth_AUTN</i>	UE authenticated HSS by AUTN properly
<i>authentication_on_ue_mme_auth_HMVP</i>	UE authenticated MME by HMVP properly
<i>authentication_on_mme_hss_auth_HNID</i>	MME authenticated HSS by HNID properly
<i>authentication_on_mme_ue_msg_auth_NNK</i>	MME authenticated UE's message by NNK properly
<i>authentication_on_hss_ue_msg_auth_NNK</i>	HSS authenticated UE's message by NNK properly
<i>secrecy_of_ue_mme_IMSI</i>	IMSI is sent from UE node to MME safely
<i>secrecy_of_ue_mme_HIMSI</i>	HIMSI is sent from UE node to MME safely
<i>secrecy_of_ue_mme_HRu</i>	HRu is sent from UE node to MME safely
<i>secrecy_of_ue_mme_Kum</i>	Kum is sent from UE node to MME safely
<i>secrecy_of_ue_mme_HMVP</i>	HMVP is sent from UE node to MME safely
<i>secrecy_of_ue_mme_RES</i>	RES is sent from UE node to MME safely
<i>secrecy_of_ue_hhs_UIH</i>	UIH is sent from UE node to HSS safely
<i>secrecy_of_ue_hhs_HMVP</i>	HMVP is sent from UE node to HSS safely
<i>secrecy_of_ue_hhs_HIMSI</i>	HIMSI is sent from UE node to HSS safely
<i>secrecy_of_mme_hss_kmh</i>	Kmh is sent from MME node to HSS safely
<i>secrecy_of_mme_hss_HHVP</i>	HHVP is sent from MME node to HSS safely
<i>secrecy_of_mme_hss_HSNID</i>	HSNID is sent from MME node HSS safely
<i>secrecy_of_mme_hss_UIH</i>	UIH is sent from MME to HSS securely
<i>secrecy_of_mme_hss_HVP</i>	HVP is sent from MME to HSS securely
<i>secrecy_of_mme_hss_HIMSI</i>	HIMSI is forwarded from MME securely
<i>secrecy_of_mme_ue_SQN</i>	SQN is forwarded from MME to UE securely
<i>secrecy_of_mme_ue_RAND</i>	SQN is forwarded from MME to UE securely
<i>secrecy_of_mme_ue_AUTN</i>	AUTN is forwarded from MME to UE securely
<i>secrecy_of_mme_ue_MA</i>	MA is forwarded from MME to UE securely
<i>secrecy_of_mme_ue_RNK</i>	RNK is forwarded from MME to UE securely
<i>secrecy_of_mme_ue_AMF</i>	AMF is forwarded from MME to UE securely
<i>secrecy_of_hss_mme_KASME</i>	KASME is forwarded from HSS securely
<i>secrecy_of_hss_mme_XRES</i>	XRES is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_AUTN</i>	AUTN is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_RAND</i>	RAND is forwarded from HSS MME securely
<i>secrecy_of_hss_mme_SQN</i>	SQN is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_HNID</i>	HNID is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_RNK</i>	RNK is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_MA</i>	MA is forwarded from HSS to MME securely
<i>secrecy_of_hss_mme_AMF</i>	AMF is forwarded from HSS to MME securely

Figure 7 and 8 indicate the evaluation results of the REPS-AKA1 method in OFMC and CL-AtSe modes, respectively (Details are in Appendix1). The “SAFE” shows that the simulated method is safe against the automatically created attacks.

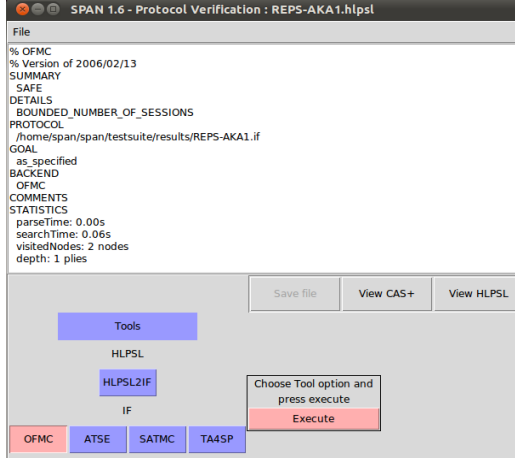


Figure 7: Evaluation result in OFMC

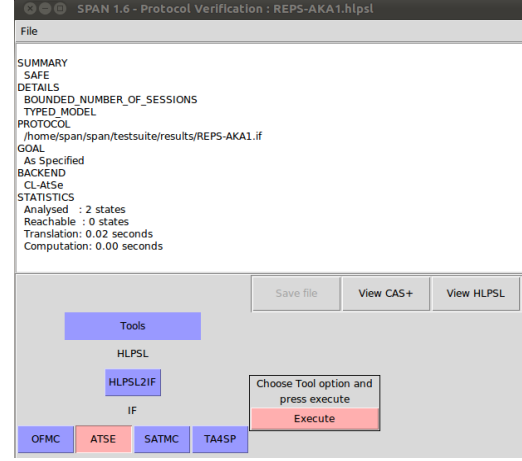


Figure 8: Evaluation result in CL-AtSe

3.2.2 Formal analysis using Scyther

The proposed REPS-AKA1 method is analyzed by means of Scyther tool which is an automated security verification tool under the cryptography assumption. This tool can demonstrate security threats for each modeled method. The presented protocol is modeled by using SPDL in the Scyther tool. The description of the proposed REPS-AKA1 protocols is accomplished according to the defined nodes in the protocol. There are three nodes (UE, HSS, and MME) in REPS-AKA1 protocol. In the modeled protocol, five claims (Nisynch, Secrecy, Weakagree, Niagree, and Alive) are evaluated for the involved nodes (Scyther codes are presented in Appendix 2).

Based on UE's viewpoint in REPS-AKA1, the security claims are presented as follows:
 $claim_ue1(UE, Secret, HMSI); claim_ue2(UE, Secret, HRu); claim_ue3(UE, Secret, HMVP); claim_ue4(UE, Secret, kum); claim_ue5(UE, Secret, NNK); claim_ue6(UE, Secret, UIH); claim_ue7(UE, Secret, HRES); claim_uea(UE, Alive); claim_uew(UE, Weakagree); claim_uenia(UE, Niagree);$

Based on MME's viewpoint in REPS-AKA1, the security claims are presented as follows:
 $claim_mme1(MME, Secret, HMSI); claim_mme2(MME, Secret, HMVP); claim_mme3(MME, Secret, UIH); claim_mme4(MME, Secret, kmh); claim_mme5(MME, Secret, HHVP); claim_mme6(MME, Se$

cret,HSNID);claim_mme7(MME,Secret,NNK);claim_mme8(MME,Secret,SQN);claim_mme9(MME,Secret,RAND);claim_mme10(MME,Secret,AUTN);claim_mme11(MME,Secret,HMVP);claim_mme12(MME,Secret,RNK);claim_mme13(MME,Secret,MA);claim_mmea(MME,Alive);claim_mmew(MME,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nisynch);

Based on HSS's viewpoint in REPS-AKA1, the security claims are presented as follows:*claim_hss1(HSS,Secret,HNID);claim_hss2(HSS,Secret,SQN);claim_hss3(HSS,Secret,RAND);claim_hss4(HSS,Secret,AUTN);claim_hss5(HSS,Secret,KASME);claim_hss6(HSS,Secret,HRES);claim_hss7(HSS,Secret,HMVP);claim_hss8(HSS,Secret,RNK);claim_hss9(HSS,Secret,MA);claim_hssa(HSS,Alive);claim_hssw(HSS,Weakagree);claim_hssnia(HSS,Niagree);claim_hssnis(HSS,Nisynch);*

The evaluation results of modeled protocol are shown in Figure 9, 10, and 11. According to these figures, the proposed protocol satisfy all defined security goals (i.e., climes) in involved nodes (UE, MME, and HSS).

Claim				Status		Comments
REPSAKA1	UE	REPSAKA1,ue1	Secret H1(IMS)	Ok	Verified	No attacks.
		REPSAKA1,ue2	Secret H2(Ru)	Ok	Verified	No attacks.
		REPSAKA1,ue3	Secret H3(MVP)	Ok	Verified	No attacks.
		REPSAKA1,ue4	Secret H5(XOR(IMS,XOR(Rk,OSK)))	Ok	Verified	No attacks.
		REPSAKA1,ue5	Secret XOR(Rk,OSK)	Ok	Verified	No attacks.
		REPSAKA1,ue6	Secret H4(XOR(IMS,OSK))	Ok	Verified	No attacks.
		REPSAKA1,ue7	Secret H12(F2(RAND,H10(H4(XOR(IMS,OSK)))))	Ok	Verified	No attacks.
		REPSAKA1,uea	Alive	Ok	Verified	No attacks.
		REPSAKA1,uew	Weakagree	Ok	Verified	No attacks.
		REPSAKA1,uenia	Niagree	Ok	Verified	No attacks.
		REPSAKA1,uenis	Nisynch	Ok	Verified	No attacks.

Figure 9: Scyther evaluation for defined claims in the UE (REPS-AKA1)

Claim				Status		Comments
REPSAKA1	MME	REPSAKA1,mme1	Secret H1(IMSI)	Ok	Verified	No attacks.
		REPSAKA1,mme2	Secret H3(MVP)	Ok	Verified	No attacks.
		REPSAKA1,mme3	Secret H4(XOR(IMSI,OSK))	Ok	Verified	No attacks.
		REPSAKA1,mme4	Secret H7(XOR(SNID,H1(IMSI)))	Ok	Verified	No attacks.
		REPSAKA1,mme5	Secret H8(HVP)	Ok	Verified	No attacks.
		REPSAKA1,mme6	Secret H6(SNID)	Ok	Verified	No attacks.
		REPSAKA1,mme7	Secret XOR(Rk,OSK)	Ok	Verified	No attacks.
		REPSAKA1,mme9	Secret RAND	Ok	Verified	No attacks.
		REPSAKA1,mme10	Secret AUTHFunc(XOR(SQN,F5(RAND,H10(H4(XOR(IMSI,OSK)))))	Ok	Verified	No attacks.
		REPSAKA1,mme11	Secret H3(MVP)	Ok	Verified	No attacks.
		REPSAKA1,mme12	Secret H10(H4(XOR(IMSI,OSK)))	Ok	Verified	No attacks.
		REPSAKA1,mme13	Secret H11(H10(H4(XOR(IMSI,OSK))),H6(SNID))	Ok	Verified	No attacks.
		REPSAKA1,mme14	Secret AMF	Ok	Verified	No attacks.
		REPSAKA1,mmea	Alive	Ok	Verified	No attacks.
		REPSAKA1,mmew	Weakagree	Ok	Verified	No attacks.
		REPSAKA1,mmenia	Niagree	Ok	Verified	No attacks.
		REPSAKA1,mmenis	Nisynch	Ok	Verified	No attacks.

Figure 10: Scyther evaluation for defined claims in the MME (REPS-AKA1)

Claim				Status		Comments
REPSAKA1	HSS	REPSAKA1,hss1	Secret H9(MVP,HVP)	Ok	Verified	No attacks.
		REPSAKA1,hss2	Secret SQN	Ok	Verified	No attacks.
		REPSAKA1,hss3	Secret RAND	Ok	Verified	No attacks.
		REPSAKA1,hss4	Secret AUTHFunc(XOR(SQN,F5(RAND,H10(H4(XOR(IMSI,OSK)))))	Ok	Verified	No attacks.
		REPSAKA1,hss5	Secret KDF(XOR(SQN,F5(RAND,H10(H4(XOR(IMSI,OSK)))))...	Ok	Verified	No attacks.
		REPSAKA1,hss6	Secret H12(F2(RAND,H10(H4(XOR(IMSI,OSK)))))	Ok	Verified	No attacks.
		REPSAKA1,hss7	Secret H3(MVP)	Ok	Verified	No attacks.
		REPSAKA1,hss8	Secret H10(H4(XOR(IMSI,OSK)))	Ok	Verified	No attacks.
		REPSAKA1,hss9	Secret H11(H10(H4(XOR(IMSI,OSK))),H6(SNID))	Ok	Verified	No attacks.
		REPSAKA1,hss10	Secret AMF	Ok	Verified	No attacks.
		REPSAKA1,hssa	Alive	Ok	Verified	No attacks.
		REPSAKA1,hssw	Weakagree	Ok	Verified	No attacks.
		REPSAKA1,hssnia	Niagree	Ok	Verified	No attacks.
		REPSAKA1,hssnis	Nisynch	Ok	Verified	No attacks.

Figure 11: Scyther evaluation for defined claims in the HSS (REPS-AKA1)

3.2.3 Security analysis by BAN logic

We prove that the presented REPS-AKA1 method is safe based on BAN Logic rules. This security logic is an approach to perform a security verification of AKA protocols. The BAN logic properly verifies the transmitted messages and checks confidentiality among involved nodes at different steps of each AKA method.

The formal messages of the proposed protocol (i.e., REPS-AKA1) are as follows:

Attach_Req_Msg(A1,A2): $UE \rightarrow MME: \{HIMSI, HRu, HMVP, Kum, \{NNK\}_{inv(Kpu)}\}_{Kpm}$ and $\{HIMSI, HMVP, UIH\}_{Kph}$, **Auth_Info_Req_Msg(A2, A3):** $MME \rightarrow HSS: \{HIMSI, HMVP, UIH\}_{Kph}$ and $\{Kmh, HHVP, HSNID, \{NNK\}_{inv(Kpu)}\}_{Kph}$, **Auth_Info_Reply_Msg(A4, A5):** $HSS \rightarrow MME: \{HNID, SQN, RAND, AUTN, KASME, HRES, XHMVP\}_{Kmh}$ and $\{RNK, MA, AMF\}_{Kpu}$, **Auth_Req_Msg(A6, A7):** $MME \rightarrow UE: \{SQN, RAND, AUTN, HMVP\}_{Kum}$ and $\{RNK, MA, AMF\}_{Kpu}$, **Auth_Reply_Msg(A8):** $UE \rightarrow MME: \{HRES\}_{Kum}$.

We need to transform all messages into idealized logical format as follows:

Attach_Req_Msg (A1, A2): $UE \rightarrow MME: MME \triangleleft \{UE \mid \Rightarrow \{HIMSI, HMVP, Kum, \{NNK\}_{Kpu^{-1}}, \#(HRu)\}_{Kpm} \text{ and } \{UE \mid \Rightarrow \{HIMSI, HMVP, UIH\}\}_{Kph}, \text{Auth_Info_Req_Msg (A2, A3): } MME \rightarrow HSS: HSS \triangleleft \{UE \mid \Rightarrow \{HIMSI, HMVP, UIH\}\}_{Kph} \text{ and } \{MME \mid \Rightarrow \{Kmh, HHVP, HSNID\}, UE \mid \Rightarrow \{\{NNK\}_{Kpu^{-1}}\}\}_{Kph}, \text{Auth_Info_Reply_Msg(A4,A5): } HSS \rightarrow MME: MME \triangleleft \{HSS \mid \Rightarrow \{HNID, AUTN, KASME, HRES, XHMVP\}, \#(SQN), \#(RAND)\}_{Kmh} \text{ and } \{HSS \mid \Rightarrow \{RNK, MA, AMF\}\}_{Kpu}, \text{Auth_Req_Msg(A6, A7): } MME \rightarrow UE: UE \triangleleft \{\#(SQN), \#(RAND), HSS \mid \Rightarrow \{AUTN, HMVP\}\}_{Kum} \text{ and } \{HSS \mid \Rightarrow \{RNK, MA, AMF\}\}_{Kpu}, \text{Auth_Reply_Msg(A8): } UE \rightarrow MME: MME \triangleleft \{UE \mid \Rightarrow HRES\}_{Kum}.$

The following initial Assumptions (A) are considered to evaluate the proposed protocol: **A1:** we assume that some public keys in REPS-AKA1 (i.e. Kpu, Kpm and Kph) are used between involved entities as follows: $UE \models \xrightarrow{K_{ph}} HSS \wedge \xrightarrow{K_{pu}} UE \wedge \xrightarrow{K_{pm}} MME,$

$MME \models \xrightarrow{K_{ph}} HSS \wedge \xrightarrow{K_{pu}} UE \wedge \xrightarrow{K_{pm}} MME$, $HSS \models \xrightarrow{K_{ph}} HSS \wedge \xrightarrow{K_{pu}} UE$, **A2:** It is considered that some secret keys in REPS-AKA1 (i.e. Kum, OSK, and Kmh) is used between different nodes as follows: $UE \models UE \xleftrightarrow{K_{um}} MME \wedge HSS \xleftrightarrow{OSK} UE$, $MME \models UE \xleftrightarrow{K_{um}} MME \wedge MME \xleftrightarrow{K_{mh}} HSS$, $HSS \models HSS \xleftrightarrow{OSK} UE \wedge MME \xleftrightarrow{K_{mh}} HSS$, **A3:** It is assumed that the entities calculate (i.e., control) some security parameters and generated keys as follows: $UE \models NNK \wedge HIMSI \wedge HRu \wedge HMVP \wedge UIH \wedge Kum \wedge XRNK \wedge XMA \wedge XMAC \wedge RES \wedge CK \wedge IK \wedge AK \wedge KASME \wedge AUTN \wedge HRES$, $MME \models HSNID \wedge Kmh \wedge Kum \wedge HHVP \wedge HNID$, $HSS \models HIMSI \wedge XHHVP \wedge XHMVP \wedge XUIH \wedge RNK \wedge MA \wedge HNID \wedge MAC \wedge XRES \wedge CK \wedge IK \wedge AK \wedge HRES \wedge KASME \wedge AUTN$, **A4:** We assume that the entities generate some random value to use in key derivation and authentication functions as follows: $UE \models \#(RNue) \wedge \#(Ru)$, $HSS \models \#(RAND)$, **A5:** It is considered that the entities have some security value related to other involved nodes as follows: $UE \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge (UE \xleftrightarrow{SQN} HSS) \wedge (UE \xleftrightarrow{MVP} HSS) \wedge (UE \xleftrightarrow{MVP} MME)$, $MME \models (MME \xleftrightarrow{HVP} HSS) \wedge (MME \xleftrightarrow{MVP} HSS) \wedge (UE \xleftrightarrow{MVP} MME) \wedge (MME \xleftrightarrow{SNID} HSS)$, $HSS \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge (UE \xleftrightarrow{SQN} HSS) \wedge (UE \xleftrightarrow{MVP} HSS) \wedge (MME \xleftrightarrow{SNID} HSS) \wedge (MME \xleftrightarrow{MVP} HSS)$.

The main security Goals (G) of the suggested protocol are represented as follows: **G1:** The mutual authentication between involved nodes is ensured in the method as $UE \models HSS \models MME \wedge MME \models HSS \models UE$ then $UE \models MME \wedge MME \models UE$, **G2:** Key acceptance achieved between all defined entities. There is a matched KASME between the UE, HSS, and MME to run the proper key agreement. As $MME \models UE \models KASME \wedge UE \models MME \models KASME$ then $UE \xleftrightarrow{KASME} MME$, thus, $UE \xleftrightarrow{KASME} HSS$, **G3:** Message authentication between corresponding entities as follows: $MME \models UE \sim NNK$ (in Attach_Req_Msg (A1, A2)), $HSS \models UE \sim NNK$ (in Auth_Info_Req_Msg (A2, A3)), **G4:** The key freshness between UE and network side (i.e., MME, and HSS nodes) and also among network nodes. The

key freshness between these entities holds since $UE \models HSS \models \#(Ru) \wedge \#(RAND)$, $HSS \models UE \models \#(Ru) \wedge \#(RAND)$, thus, CK, IK, AK, and KASME are fresh keys accordingly.

We evaluate the idealized form of presented method according to the hypothesis we defined and the BAN Logic rules including Message meaning rule (R1), Jurisdiction rule (R2), and Nonce Verification Rule(R3). We employ the BAN logic rule (R2) to

derive: $\frac{UE \models HSS \Rightarrow AUTN, UE \models HSS \models AUTN}{UE \models AUTN}, \frac{HSS \models UE \Rightarrow AUTN, HSS \models UE \models AUTN}{HSS \models AUTN},$

$\frac{HSS \models UE \Rightarrow UIH, HSS \models UE \models UIH}{HSS \models UIH}, \frac{UE \models HSS \Rightarrow UIH, UE \models HSS \models UIH}{UE \models UIH}$ To satisfy G1. We utilize the BAN

logic rule (R1) to derive: $\frac{MME \models (MME \xrightarrow{K_{mh}} HSS), MME \triangleleft \{KASME\}_{K_{mh}}}{MME \models HSS \models KASME} \wedge HSS \models KASME$ To satisfy

G2. We use the BAN logic rule (R1) to derive:

$\frac{MME \models (\frac{PK_{ue}}{PK_{ue}} UE), MME \triangleleft \{NNK\}_{PK_{ue}^{-1}}}{MME \models UE \models NNK}, \frac{HSS \models (\frac{PK_{ue}}{PK_{ue}} UE), HSS \triangleleft \{NNK\}_{PK_{ue}^{-1}}}{HSS \models UE \models NNK}$ To satisfy G3. We employ the

BAN logic rule (R3) to derive:

$\frac{UE \models \#(RAND), UE \models HSS \models \sim RAND}{UE \models HSS \models RAND} \wedge \frac{UE \models HSS \models RAND, UE \models HSS \models RAND}{UE \models RAND}$. Thus, all other

keys (i.e., CK, IK, AK, and KASME) are generated using fresh values. We use the same approach to prove key freshness in other entities. Thus, the protocol satisfies G4.

3.2.4 Proof of security requirements

The correctness proofs for the presented protocol (i.e., REPS-AKA1) based on security requirements. The evaluation results of the REPS-AKA1 are as follows:

Claim 1: Strong message authentication is provided in REPS-AKA1.

Proof: The REPS-AKA1 method satisfy the message authentication in MME and HSS entities by using NNK value. Accordingly, they can verify that the received message is from a valid user node (i.e., UE). This property can reduces the probability of future attacks such as DoS/DDoS attacks in LTE systems.

Claim 2: There is a robust mutual authentication between involved nodes.

Proof: The proposed method provides robust authentication between involved nodes in AS/NAS layers. The UE can confirm the HSS and MME nodes by means of AUTN and HMVP, respectively. And MME verifies UE and HSS by HRES and HNID properly. The HSS ensures robust authentication between the UE and MME. In REPS-AKA1, the UE verifies the HSS using the AUTN. The HSS can authenticate the UE using the UIH. The HSS also verifies the MME using the HMVP.

Claim 3: Protection of UE's Identity (i.e., IMSI) is provided in REPS-AKA1.

Proof: In standard EPS-AKA, the MME requests the IMSI value from the user, UE forwards the IMSI value to the MME. Therefore, an adversary can easily achieve the IMSI of UE and launch different malicious attacks. In REPS-AKA1 approach, the UE hashes the IMSI parameter and forwards the encrypted user's identity to the LTE network. It also uses secret keys (i.e. K_{um} and K_{mh}) to decrease computational cost and signaling overload. Only the HSS has an original key (i.e., OSK) and can extract the IMSI value of the UE. Thus, HSS node can use the user's identity securely.

Claim 4: The protocol ensures resistance to key leaked.

Proof: The UE's AV transmission is secure by using the K_{mh} . In addition, the HSS authenticates the MME successfully before sending the AVs. Thus, a rogue MME needs to be authenticated successfully in the system. The attacker must know the valid MME's private key to have these vectors, which is impossible in REPS-AKA1 method.

Claim 5: The REPS-AKA1 ensures resistance to MitM attack.

Proof: The method can be resistant to the MitM attack properly. A malicious attacker cannot extract the UE's identity since the IMSI is protected using a generated secret key. The other information such as UIH, HIND, HSNID, and AUTN is not sent as a plaintext message. Accordingly, REPS-AKA1 is safe against MitM attacks.

Claim 6: The REPS-AKA1 protocol is resistant to Brute force attack.

Proof: In the Brute force attacks, an attacker can reach to main keys by checking all possible keys. In REPS-AKA1 method, the OSK and generated keys are produced in an appropriate way that makes them very hard to be obtained.

Claim 7: The protocol achieves optimization of Key generation overload.

Proof: In the REPS-AKA1 protocol, the Kum and Kmh keys generated in UE and MME nodes will be used for future handovers. Thus, there is no need to make them frequently in LTE entities which helps decrease key generation overload.

Claim 8: The REPS-AKA1 is resistant to SNID leakage.

Proof: The HSS can verify the MME by using the MVP parameter. In REPS-AKA1, the HSS protects the MME from being impersonated even if its SNID is disclosed.

Claim 9: The REPS-AKA1 method is resistant to Single-key issues.

Proof: The protocol ensures resistance to this security issue. The proposed approach uses a dynamic key generation process instead of the permanent one. In the REPS-AKA1, RNK key linked with OSK and IMSI is used to make all AKA keys. Moreover, the UE and HSS generate a fresh RNK key and derived KASME keys for each fresh AKA procedure. Thus, an adversary cannot obtain any User's data using insecure keys.

Claim 10: The protocol provides confidentiality in control plan.

Proof: By sharing asymmetric and symmetric keys between UE, MME, and HSS, the proposed REPS-AKA1 approach provides perfect confidentiality to all LTE interfaces. Therefore, this protocol can prevent DoS and MitM attacks, which are workable because of the unprotected transmission of AKA parameters.

Claim 11: The REPS-AKA1 provides prevention of Replay attacks.

Proof: The REPS-AKA1 can prevent Replay attack suitably. The UE uses a fresh HRu value. The HRu is sent to MME node safely. The MME confirms the request message based on received random value upon receiving attach request. If it is a fresh HRu, the MME verifies this received message; if not, the MME reject that request message.

Claim 12: The REPS-AKA1 provides resistance to redirection attack.

Proof: To provide protection against redirection attacks, the proposed method uses the challenge-response procedure. The HSS verifies the UE's authenticity and the serving MME using received UIH and HMVP. For verifying a valid HSS, the UE similarly computes AUTN value and matches it with the received one. The UE similarly

computes RES for UE authentication as HSS computes XRES. The MME checks whether $RES = XRES$ is satisfied or not. Thus, the proposed REPS-AKA1 prevents the deployment of rogue LTE entities, which mitigates the redirection attack.

Claim 13: The REPS-AKA1 provides prevention of DoS attack.

Proof: In REPS-AKA1 protocol, MME checks Ru and HMSI in each authentication process. Thus, an adversary cannot send many attach requests from a fake UE in a given interval. Also, there is no need to ask for all authentication parameters from HSS since MME can authenticate UE by itself, and it can help the system to decrease the LTE traffic load properly. Thus, REPS-AKA1 is robust to DoS attacks.

Claim 14: The protocol achieves the prevention of Impersonation attack.

Proof: In REPS-AKA1 protocol, it is not feasible for an attacker to run this type of attack by capturing and modifying the transmitted parameters. Because the adversary without the OSK cannot generate legal access to the UE and LTE nodes. The UE's identity is protected by hashing and encryption. Thus, the attacker never finds the IMSI and fails to perform the impersonation attacks.

In Table 4, the proposed REPS-AKA1 protocol is compared with some other suggested AKA protocols based on the different security features [10] [42]. This comparison allows us to place REPS-AKA1 in the group of robust AKA approaches in the LTE system.

Table 4: Evaluation of the presented AKA methods

AKA Security Features	EPS-AKA	EC-AKA [45]	SE- EPS-AKA [46]	MEPS AKA [26]	M- LTE-AKA [25]	REPS- AKA1
Secure against IMSI Leak	No	Yes	Yes	Yes	Yes	Yes
Secure against MitM attack	No	Yes	Yes	Yes	Yes	Yes
Secure against DoS attack	No	No	No	No	No	Yes
Secure against redirection attack	No	No	No	No	No	Yes
Secure against impersonation attack	No	Yes	Yes	Yes	Yes	Yes
Secure against replay attack	No	No	Yes	Yes	No	Yes
Strong mutual authentication	No	No	No	No	No	Yes
Independency of Single secret key	No	No	No	Yes	No	Yes
Secure against HSS overloading	Yes	Yes	Yes	No	Yes	No
Data confidentiality in AKA	No	Yes	No	No	No	Yes
Total number of satisfied features	1	5	5	5	4	9

3.3 Performance analysis of the proposed model: REPS-AKA1

In this section, an evaluation of suggested REPS-AKA1 protocol is presented. The analysis clarifies that the REPS-AKA1 method can satisfy the QoS requirements efficiently. The performance analyses are carried out based on some performance metrics such as bandwidth consumption, computational cost, and signaling overhead.

3.3.1 Bandwidth consumption (core traffic)

We employ a common traffic model to evaluate the bandwidth consumption between MME and HSS entities. The traffic model of AKA protocol is proportional to some parameters as shown in Table 5. The “ L ”, “ MD ”, “ V ”, and “ Ar ” are Length of the registration cell, Mean Density of user nodes (i.e., UE), Velocity of the UE, and total number of registration areas respectively [12, 24, 41]. This model is applicable for LTE systems. Assumption values for traffic model are shown in Table 5.

Table 5: Assumption values for Traffic’s model

Parameters	Assumption value
Ar	100
L	25.3 km
MD	350 / km ²
V	4.8 km/h

If we assume that the direction of user's movement is spread over $[0-2\pi]$, the Rate of user's authentication request (R) is defined by (3.1) as:

$$R = (MD \times V \times L)/\pi \quad (3.1)$$

Then Total generated Requests (TR) can be measured as:

$$TR = R \times Ar \quad (3.2)$$

The bandwidth consumption between the MME and the HSS (BW_{mh}) can be measured as [12, 24, 41]:

$$BW_{mh} = \sum_{k=1}^{k=n} (MSG)_k \times TR \quad (3.3)$$

Where “ n ” shows the number of transmitted messages between MME and HSS, and $(MSG)_k$ presents the length of k^{th} message (in bits). Based on Table 5, the TR is calculated ($TR=376$). According to (3.2) and (3.3), we compute the BW_{mh} of the REPS-AKA1 and compare it with two other protocols including the standard EPS-AKA method and Modified LTE-AKA (MLTE-AKA) as follows [12, 24, 41]:

The BW_{mh} of EPS-AKA: In the EPS-AKA protocol, two authentication messages are used between the MME and the HSS entities. The MME transmits a request to the HSS node via $MSG1$. Then, the HSS in home network sends “ n ” number of AVs to the serving network via $MSG2$ message. Therefore, the number of transmitted bits is presented as: $MSG1 = (IMSI) + (SNID) = 128 + 48 = 176 \text{ bits}$, $MSG2 = AV\{(XRES) + (AUTH) + (RAND) + (K_{ASME})\}_n = 544n \text{ bits}$, $BW_{mh} = (176 + 544n) \times TR$. According to (3.3) and Table 5, when the number of AVs is 7 (i.e., $n=7$), the BW_{mh} of EPS-AKA protocol is expressed as 1.49 Mbits/s.

The BW_{mh} of MLTE-AKA: In the MLTE-AKA method, $MSG1$ and $MSG2$ are forwarded between involved nodes. Thus, the number of transmitted bits in the EPC part of the LTE network for MLTE-AKA can be calculated as: $MSG1 = 2 \times (Uid) +$

$(SN_{ID}) = 176 \text{ bits}$, $MSG2 = AV\{(XRES) + (AUTH) + (IMSI) + (RAND) + (K_{ASME}) + (SK)\}_n = 800n \text{ bits}$, $BW_{mh} = (176 + 800n) \times TR$. Therefore, according to (3.3), and Table 5, the BW_{mh} of MLTE-AKA protocol can be found as 2.17 Mbits/s (when $n=7$).

The BW_{mh} of the REPS-AKA1: In the REPS-AKA1, we require two messages (i.e., $MSG1$, and $MSG2$) transmitted among the MME and the HSS in EPC part of LTE network. The total number of transmitted bits between these nodes can be found as:

$$\begin{aligned} MSG1 &= (HIMSI) + (HMVP) + (UIH) + (Kmh) + (HHVP) + (HSNID) + (NNK) = 128 + \\ &128 + 128 + 128 + 128 + 128 + 128 = 896 \text{ bits}, MSG2 = (HNID) + (SQN) + (RAND) + \\ &(AUTH) + (K_{ASME}) + (HRES) + (XHMVP) + (RNK) + (MA) + (AMF) = 128 + 48 + 128 + \\ &128 + 256 + 128 + 128 + 128 + 128 + 16 = 1216 \text{ bits}, \quad BW_{mh} = (896 + 1216) \times TR. \end{aligned}$$

Consequently, using (3.3) with the parameters in Table 5 (when $n=7$), the BW_{mh} of the REPS-AKA1 can be found 0.79 Mbits/s.

Thus, the achieved improvements of the REPS-AKA1 method over the EPS-AKA and MLTE-AKA are 46% and 63% respectively. Then, we compare these AKA methods according to different authentication vector sizes. The evaluation results are depicted in Figure 12. The evaluation results present that the BW_{mh} of the suggested method is fixed. However, there is a significant growth in BW_{mh} of other methods in particular when value of “ n ” (i.e., AV) increases.

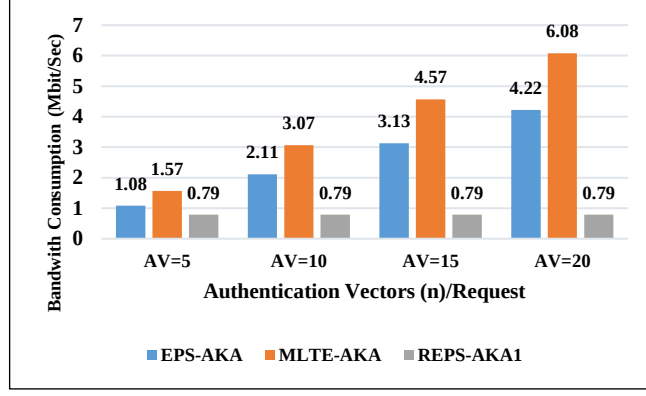


Figure 12: Evaluation of bandwidth consumption in core traffic

The performance of AKA methods can be different according to the MD values. Figure 13 shows behavior of protocols based on different MD values (when $AV=7$). As can be seen, BW_{mh} of all methods increases as MD increases, however, REPS-AKA1 outperforms the other two protocols as it has the lowest BW_{mh} and the smallest slope.

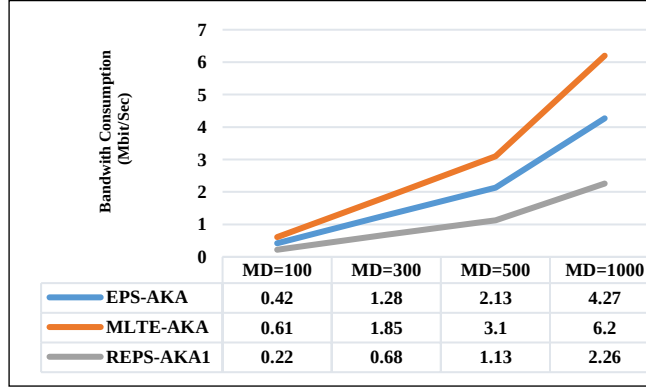


Figure 13: Bandwidth consumption of core traffic (when $n=7$)

Some methods have better BW_{mh} (e.g., Ref. [63]= $1088 \times TR$, Ref. [25]= $1248 \times TR$).

3.3.2 Computational cost

We check security functions for Computational Cost (CC). Each function's cost is supposed to be 1. Thus, the CC of each protocol is calculated as shown in (3.4) below

$$[12, 24, 41]: CC = \sum_{k=1}^{k=m} Cost(node)_k \times TR \quad (3.4)$$

Where “ m ” is the number of nodes (i.e., $m = 3$: UE, MME, and HSS) in the system.

Here, we compare the CC of the REPS-AKA1 method with some others as follows:

The CC of EPS-AKA protocol: Total security functions involved in EPS-AKA can be expressed as: $UE_{cost} = (f1 + f2 + f3 + f4 + f5 + 2KDF) = 7$, $MME_{cost} = (No\ security\ function) = 0$, $HSS_{cost} = AV(f1 + f2 + f3 + f4 + f5 + 2KDF)n = 7n$, (n = number of AVs). The CC of EPS-AKA can be as: $CC_{EPS-AKA} = TR \times (7 + 7n)$.

The CC of MLTE-AKA protocol: Total security functions involved in the MLTE-AKA method are expressed as: $UE_{cost} = (Enc + 2fk + MAC + f1 + f2 + f3 + f4 + f5 + 3KDF) = 12$, $MME_{cost} = (No\ security\ function) = 0$, $HSS_{cost} = Dec + AV(3KDF + f1 + f2 + f3 + f4 + f5)n = 1 + 8n$. Accordingly, by using equation (3.4), the CC of the MLTE-AKA can be calculated as: $CC_{MLTE-AKA} = TR \times (13 + 8n)$.

The CC of REPS-AKA1 protocol: Total number of security functions used in suggested REPS-AKA1 approach are as follows: $UE_{cost} = (F_A + F_S + H1 + H2 + H3 + H4 + H5 + H10 + H11 + H12 + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc) = 17$, $MME_{cost} = (F_A + F_S + H6 + H7 + H8 + H9) = 6$, $HSS_{cost} = (F_A + F_S + H1 + H3 + H4 + H8 + H9 + H10 + H11 + H12 + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc) = 17$, Where F_A and F_S are Symmetric and Asymmetric encryption respectively. Hence, using (3.4) the CC of the REPS-AKA1 protocol is calculated as: $CC_{REPS-AKA1} = TR \times (40)$.

Figure 14 shows the CC of AKA methods based on different value of “ n ”. The suggested method has lower CC compare to the EPS-AKA and MLTE-AKA methods (when $TR = 376$). The average computational cost improvement of REPS-AKA1 over EPS-AKA and MLTE-AKA is 71% and 76%, respectively.

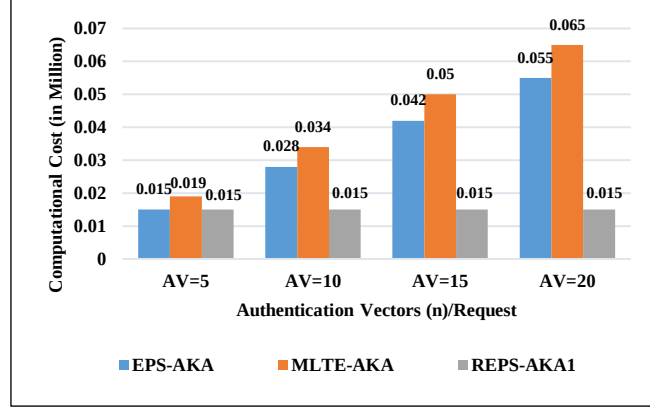


Figure 14: Computational cost in AKA protocols

We compare the REPS-AKA1 method with other two protocols based on different MD (when $n = 7$, and $TR = 376$). As can be seen in Figure 15, the CC of all AKA methods rises as MD increases. However, the REPS-AKA1 approach surpasses others since the proposed method has the lowermost CC and the slightest slope.

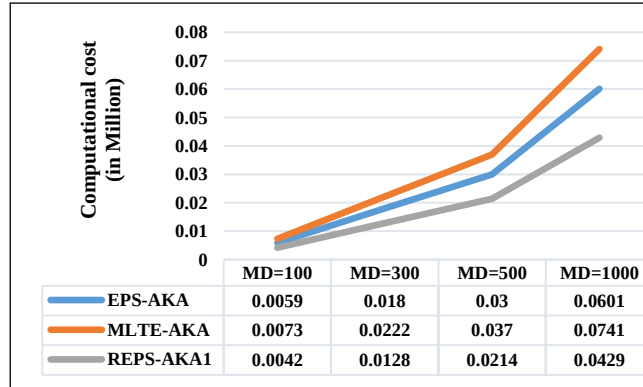


Figure 15: Computational cost in AKA protocols (when $n=7$)

3.3.3 Signaling overhead

The steps of AKA protocol in the REPS-AKA1 approach are equal to the AKA steps in the existing EPS-AKA scheme. Accordingly, there are no more round-trip delays in the REPS-AKA1 protocol. Moreover, Table 6 shows the Signaling Overload (SO) of other AKA protocols [42] [51-60][71]. The evaluation results illustrate that REPS-AKA1 has less SO compare to other AKA protocols.

Table 6: Comparison of *SO* metric

AKA method	Number of Message	AKA method	Number of Message
<i>EG-AKA</i>	12	<i>GR-AKA</i>	7
<i>GBAAM-AKA</i>	10	<i>REPS-AKA1</i>	6
<i>GROUP-AKA</i>	9	<i>EPS-AKA</i>	6
<i>GLARM-AKA/ PRIVACY-AKA /SEGB-AKA/NOVEL-AKA/ SE-AKA</i>	8		

3.3.4 Storage overhead in the serving network

The Storage Overhead (*StO*) of MME node in various AKA protocols is evaluated. In the REPS-AKA1 protocol, the MME uses previous security parameters to generate new AVs. However, in other AKA methods the MME have to keep “*n*” AVs. The comparison of the *StO* for different AKA methods is presented in Table 7.

Table 7: The *StO* of different AKA protocols

AKA methods	<i>StO</i> (bits)	AKA methods	<i>StO</i> (bits)
<i>Ref. [10]</i>	392n	<i>Ref. [62]</i>	608n
<i>Ref. [24]</i>	640n	<i>Ref. [63]</i>	704n
<i>Ref. [61]</i>	448n	<i>Ref. [25]</i>	800n
<i>Ref. [29]</i>	432n	<i>Proposed REPS-AKA1</i>	1184

According to Table 7, the proposed protocol can reduce the *StO* of the MME node.

Figure 16 present a comparison of *StO* between different AKA protocols.

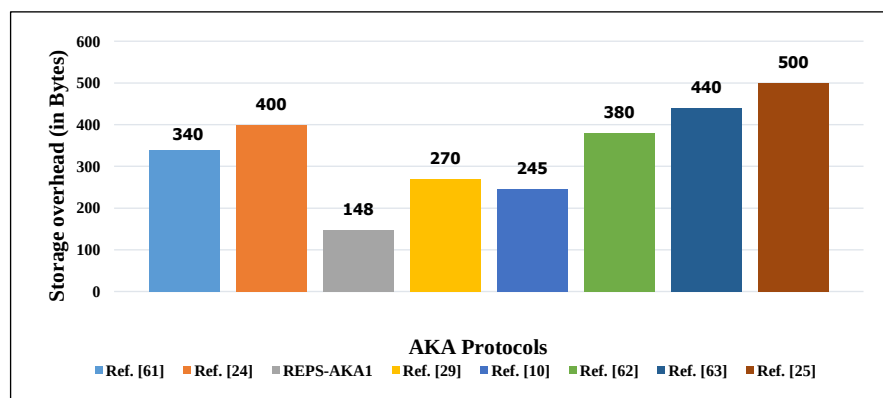


Figure 16: Storage overhead in AKA protocols (when *n*=5)

According to extracted results, the proposed method has the lowermost *StO* compare to other mentioned methods. The *StO* of the suggested method is fixed, but, there is an extensive growth in the *StO* of other methods.

The storage overheads of suggested protocols are calculated based on four AV's size (when "*n*" is 5, 10, 15, and 20). The average improvements of proposed protocol over EPS-AKA, Ref. [24], Ref. [61], Ref. [29], Ref. [10], Ref. [62], Ref. [63], and Ref. [25] are 82%, 85%, 78%, 78%, 75%, 84%, 86%, and 88% respectively.

Chapter 4

MODEL 2 (REPS-AKA2)

In this chapter, we present the REPS-AKA2 method which is an efficient way to update session keys during the handover procedure. The main goal is to make optimized key management and update according to a few key parameters. These optimal parameters and updating keys mechanism improve the communication overhead in Access Stratum (AS) and Non-Access Stratum (NAS) layers properly. We also evaluate our method according to security features by AVISPA, Scyther tools, and BAN logic. This chapter is organized as follows: In section 4.1, the REPS-AKA2 protocol is presented in detail. Section 4.2 evaluates the proposed protocol by means of AVISPA tools, BAN Logic, and Scyther tools. The performance evaluations of the REPS-AKA2 approach are provided in section 4.3.

4.1 The proposed REPS-AKA2 protocol

We suggested a robust authentication protocol (REPS-AKA2) to settle the security vulnerabilities in the X2-Handover. We presented an improved key management model that covers forward/backward key separation efficiently. For efficiency reasons, the key management during the X2-Handover procedure does not involve the HSS node. The proposed scheme improves the existing X2-Handover, consisting of two main phases including initial authentication phase (i.e., phase 1) and re-authentication phase (i.e., phase 2).

The steps of the REPS-AKA in phase 1 are as follows (Figure 17):

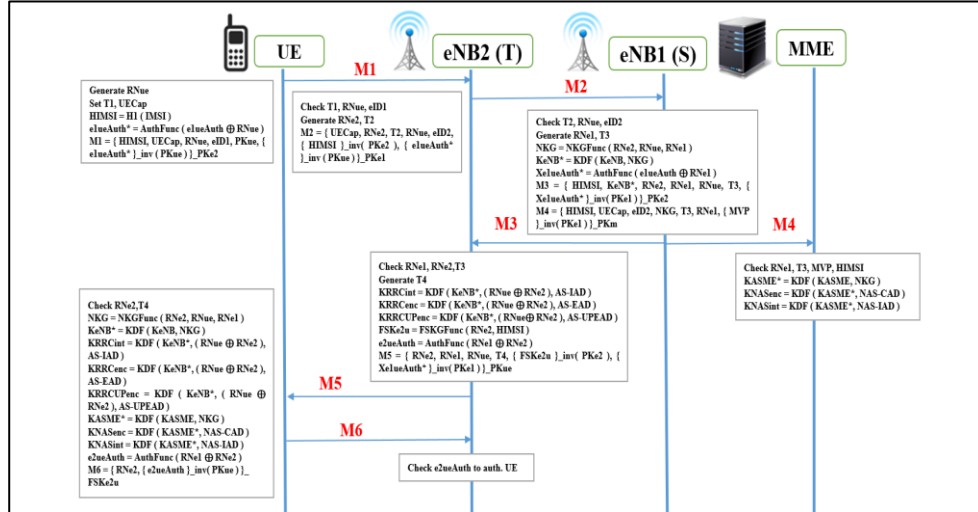


Figure 17: Steps of REPS-AKA2 (Ph1)

1. UE $\rightarrow$ eNB2 (Target eNB): In Attach request of handover (M1), the user node (i.e., UE) produces a new random value (RNue) and adjusts its UECap value, which displays capability of encryption/decryption for that UE node. Then, we use a hash function to calculate the Hash of IMSI (HIMSI) and set T1 as a timestamp. In addition, the UE generates new e1ueAuth (e1ueAuth*) using RNue and last e1ueAuth from previous handover. The e1ueAuth is an authentication parameter between UE and eNodeB1. These nodes save the e1ueAuth after successful authentication. We encrypt e1ueAuth* by inverting the public key of UE before transition, and the target eNodeB forwards it to Source eNodeB. In this manner, Source eNodeB will be sure that this message is from a valid UE. Because the only valid UE can create that message. As a result, we can decrease the probability of DoS attacks in Source eNodeB. Finally, the UE transmits M1 including HIMSI, UECap, RNue, eID1 (identifier of eNB1), PKue (UE public key), and {e1ueAuth*}_inv(PKue) encrypted by PKe2 (public key of target eNodeB).

2. eNB2 (Target eNB) → eNB1 (Source eNB): Target eNB extract HIMSI, RNue, eID1, and PKue from M1. First of all, eNB2 will check that if RNue is fresh to protect the system against replay attack. Then, it uses eID1 to learn source eNB and saves the public key of UE in its database for future usage. Target eNB encrypts HISMI by inverting the public key of eNB2 before transition. Then, target eNodeB will forward it to Source eNodeB. In this manner, Source eNodeB will be sure that this message is from valid eNB2 because the only valid eNB2 can create that message, and we can decrease the probability of DoS attacks in Source eNodeB. Finally, eNB2 will generate RNe2, T2 and transmit M2 to eNB1 (Source eNB) securely encrypted by the eNB1 public key (PKe1). The M2 includes UECap, RNe2, T2, RNue, eID2, {HIMSI}_inv(PKe2), and {e1ueAuth*}_inv(PKue).

3. eNB1 (Source eNB) → eNB2 (Target eNB) and MME: Here is the main advantage of our protocol. The eNB1 plays the most important role in REPS-AKA2 to decreases the computational cost of AKA procedure. The eNB1 generates pair (RNe1, T3) and calculates New Key Generator (NKG) using the NKGFunc function and random values (RNe2, RNue, and RNe1); $NKG = NKGFunc(RNe2, RNue, RNe1)$. Then, eNB1 calculates $KeNB^*$ which is new session key for Target eNodeB; $KeNB^* = KDF(KeNB, NKG)$. The eNB1 also computes $Xe1ueAuth^*$ which is new expected e1ueAuth; $Xe1ueAuth^* = AuthFunc(e1ueAuth \oplus RNe1)$ and forwards it to target eNB. The source eNB encrypts the $Xe1ueAuth^*$ by the inverse of eNB1's public key to verify that the received message is sent from a valid eNB1; $\{Xe1ueAuth^*\}_inv(PKe1)$. The eNB1 also encrypts pre-shared MVP (MME Validation Parameters) using invert of eNB1 public key; $\{MVP\}_inv(PKe1)$. Accordingly, the MME node can be sure that the message is forwarded from a valid

eNB1. The MME can verify eNodeB using MVP values. The eNB1 transmits M3 including HIMSI, KeNB*, RNe2, RNe1, RNue, T3, and {Xe1ueAuth*}_inv(PKe1) encrypted by public key of eNB2 (PKe2) to eNB2 and at the same time, sends M4 to MME. The M4 contains HIMSI, UECap, eID2, NKG, T3, RNe1, and {MVP}_inv(PKe1) encrypted by public key of MME (PKm).

4. eNB2 (Target eNB) → UE: When the eNB2 receives M3 from eNB1 and extracts the message, it checks RNe1 for freshness (for replay attack). The eNB2 makes FSKe2u as symmetric key between itself and UE for future: FSKe2u=FSKGFunc(RNe2, HIMSI). The eNB2 encrypts FSKe2u by invert public key of eNB2 before transition. Therefore, the UE will be sure that this message is from a valid eNB2. In addition, the eNB2 has new session key (KeNB*), and it can calculate KRRCint, KRRCenc, and KRRUPenc by using received UECap. In fact, AS-IAD, AS-EAD, and AS-UPEAD are AS-Integrity-Algorithm-Distinguisher, AS-Encryption-Algorithm-Distinguisher, and AS-Uplink-Algorithm-Distinguisher respectively which are selected based on UECap. The eNB2 also creates e2ueAuth as authentication parameter for next handover in future; e2ueAuth=AuthFunc(RNe1 $\oplus$ RNe2). Finally, eNB2 sends RNe2, RNe1, RNue, T4, {FSKe2u}_inv(PKe2), and {Xe1ueAuth*}_inv(PKe1) by M5 (encrypted by PKue) to the UE.

5. MME: When the MME receives M4, it extracts HIMSI, UECap, eID2, NKG, T3, RNe1, and {MVP}_inv(PKe1). It checks the pair (RNe1, T3) for freshness and generates new KASME*; KASME*= KDF(KASME, NKG). Now, the MME can generate KNASenc and KNASint based on KASME* and NAS CAD and NAS IAD, which are NAS-Cyphering-Algorithm-Distinguisher and NAS- Integrity-Algorithm-Distinguisher, respectively. The MME sets them according to the UECap.

6. UE → eNB2 (Target eNB): The UE checks the pair (RNe2, T4) to be sure about message freshness. And extracts RNe2 and RNe1 by its public key and makes NKG using NKGFunc function (hash function); $NKG = \text{NKGFunc}(RNe2, RNue, RNe1)$. Then, the UE can generate new session key KeNB* using NKG; $KeNB^* = \text{KDF}(KeNB, NKG)$. Also, the UE makes KASME* by NKG and its previous KASME; $KASME^* = \text{KDF}(KASME, NKG)$. Additionally, all related keys, including KRRCint, KRRCenc, and KRRUPenc in AS layer and KNASenc and KNASint in NAS layer, are generated by KeNB* and KASME*. After that, it makes e2ueAuth as an authentication parameter between Target eNB and UE, used for future handover; $e2ueAuth = \text{AuthFunc}(RNe1 \oplus RNe2)$. Finally, The UE forwards e2ueAuth, which is encrypted by inversing its public key PKue for message authentication procedure. Message M6 is encrypted by the received secret key (FSKe2u).

7. eNB2 (Target eNB): The Target eNB receives M6 and decrypts the message to find e2ueAuth, and compares it with the generated e2ueAuth to authenticate UE.

4.1.2 Phase2 (Ph2: re-Authentication)

The REPS-AKA2 protocol has an efficient feature that makes it different from other existing X2 handovers. In many applications, UE moves between some base stations frequently. Thus, it may need to handover between two given eNBs several times in a short interval. In such a situation, running all steps of handover is not necessary. The REPS-AKA2 has a useful solution for this situation. The steps of re-Authentication phase are as follows (Figure 18):

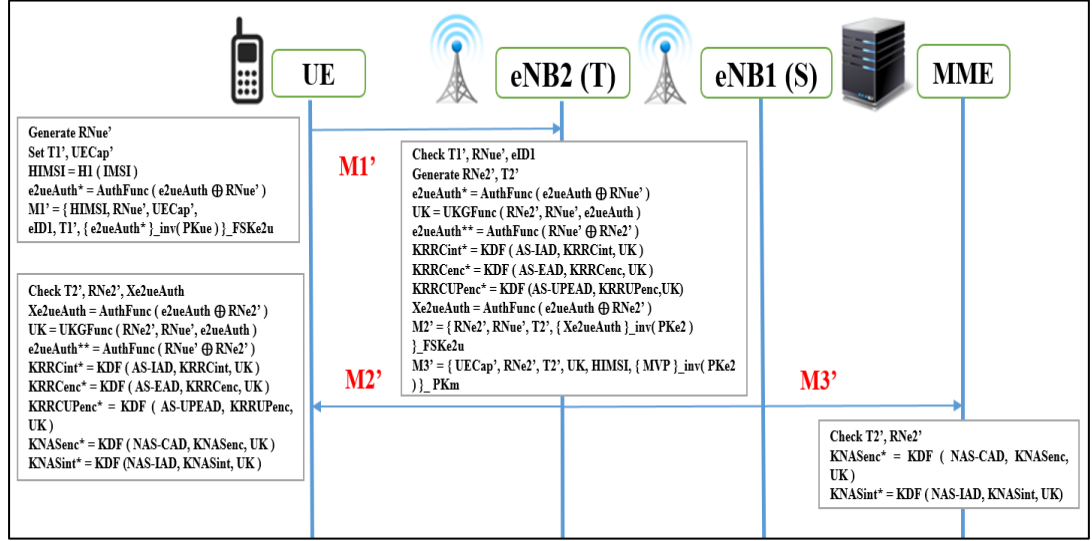


Figure 18: Steps of REPS-AKA2 (Ph2)

1. UE → eNB2 (Target eNB): In Attach request of handover (M1'), the UE generates new Random Number (RNue') and set its UEcap'. Then, we use a hash function to calculate HIMSI and set T1' as a timestamp. In addition, the UE generates e2ueAuth* (new e2ueAuth) using RNue' and last e2ueAuth from previous handover; $e2ueAuth^* = \text{AuthFunc}(e2ueAuth \oplus RNue')$. Then, the UE makes M1' including HIMSI, RNue', UEcap', eID1, T1', and $\{e2ueAuth^*\}_{inv(PKue)}$ encrypted by FSKe2u (secret key between eNB2 and UE) and forwards it to eNB2.

2. eNB2 (Target eNB) → UE and MME: Target eNB decrypts received message from UE and checks RNue' for freshness confirmation of message. Then, Target eNB computes e2ueAuth* to authenticate UE; $e2ueAuth^* = \text{AuthFunc}(e2ueAuth \oplus RNue')$. If generated e2ueAuth* matches with the received one, it authenticates UE. The Target eNB also generates new authentication parameters, e2ueAuth** for next handover; $e2ueAuth^{**} = \text{AuthFunc}(RNue' \oplus RNue2')$. Then, it generates Updated Key (UK) using RNue2', RNue', and e2ueAuth; $UK = \text{UKGFunc}(RNue2', RNue', e2ueAuth)$ to update keys and make KRRRCint*, KRRRCenc*, and KRRUCPenc*. It also computes

$Xe2ueAuth$ using $e2ueAuth$ (from pervious handover) and new $RNe2'$; $Xe2ueAuth = AuthFunc(e2ueAuth \oplus RNe2')$. After that, it sends $M2'$ including $RNe2'$, $RNue'$, $T2'$, $\{Xe2ueAuth\}_{inv(PKe2)}$ encrypted by $FSKe2u$ (secret key between $eNB2$ and UE). The $eNB2$ also forwards $M3'$ including $UECap'$, $RNe2'$, $T2'$, UK, HMSI, and $\{MVP\}_{inv(PKe2)}$ to MME. This message is encrypted by PKm.

3. MME: The MME receives $M3'$ and decrypts the message to find HMSI, $UECap'$, UK, $T2'$, $RNe1$, and $\{MVP\}_{inv(PKe2)}$. It checks the pair $(RNe2', T2')$ for freshness (replay attack) and authenticates $eNB2$ using $\{MVP\}_{inv(PKe2)}$. Then, the MME uses the UK to update related keys and make $KNASenc^*$, and $KNASint^*$. The MME generates $KNASenc^*$ and $KNASint^*$ based on the UK, previous keys ($KNASenc$ and $KNASint$), NAS-CAD and NAS-IAD.

4. UE: The UE checks the pair $(T2', RNe2')$ to be sure about message freshness. Then, UE extracts $Xe2ueAuth$ from $M2'$ and compares it with generated $Xe2ueAuth$; $Xe2ueAuth = AuthFunc(e2ueAuth \oplus RNe2')$ to authenticate $eNB2$. Then, the UE generates a new authentication parameter, $e2ueAuth^{**}$ for next handover; $e2ueAuth^{**} = AuthFunc(RNue' \oplus RNe2')$. Also, the UE computes Update Key (UK) using $RNe2'$, $RNue'$, $e2ueAuth$ (previous parameter); $UK = UKGFunc(RNe2', RNue', e2ueAuth)$. The UE uses the UK to update all keys including $KNASint^*$, $KNASenc^*$, $KRRRCenc^*$, $KRRRCint^*$, and $KRRRCUPenc^*$ in AS/NAS layers.

4.2 Security analysis of REPS-AKA2 protocol

The presented REPS-AKA2 protocol is evaluated according to different security parameters. The security evaluation of the REPS-AKA2 method is conducted by using AVISPA tool, BAN Logic, and Scyther tool.

4.2.1 Formal analysis using AVISPA

The AVISPA tool is used to verify REPS-AKA2 protocol. We employ OFMC and CL-AtSe modes to check the validation of the method. The evaluation results show that the simulated model is SAFE. The detailed security goals in REPS-AKA2 protocol are presented in Table 8. (AVISPA codes are presented in Appendix 1).

Table 8: Security goals in REPS-AKA2

Defined security goals	Description
<i>authentication_on_enbt_ue_msg_auth_e1ueAuthN</i>	eNB2(T) authenticated UE's message by e1ueAuthN properly
<i>authentication_on_ue_enbt_auth_RNue</i>	UE authenticated eNB2(T) by RNue properly
<i>authentication_on_ue_enbt_msg_auth_FSKe2u</i>	UE authenticated eNB2(T) 's message by FSKe2u properly
<i>authentication_on_enbt_ue_msg_auth_e2ueAuth</i>	eNB2(T) authenticated UE's message by e2ueAuth properly
<i>authentication_on_enbt_ue_auth_RNe2</i>	eNB2(T) authenticated UE by RNe2 properly
<i>authentication_on_enbt_ue_msg_auth_e2ueAuthN</i>	eNB2(T) authenticated UE's message by e2ueAuthN
<i>authentication_on_ue_enbt_msg_auth_Xe2ueAuth</i>	UE authenticated eNB2(T)'s message by Xe2ueAuth
<i>authentication_on_ue_enbt_auth_RNueN</i>	UE authenticated eNB2(T) by RNueN properly
<i>authentication_on_enbt_enbs_msg_auth_Xe1ueAuthN</i>	eNB2(T) authenticated eNB1(S)'s message by Xe1ueAuthN properly
<i>authentication_on_enbt_enbs_auth_RNe2</i>	eNB2(T) authenticated eNB1(S) by RNe2 properly
<i>authentication_on_enbs_enbt_msg_auth_HIMSI</i>	eNB1(S) authenticated eNB2(T)'s message by HIMSI properly
<i>authentication_on_mme_enbs_msg_auth_MVP</i>	MME authenticated eNB1(S)'s message by MVP properly
<i>authentication_on_mme_enbt_msg_auth_MVP</i>	MME authenticated eNB2(T)'s message by MVP properly
<i>secrecy_of_ue_enbt_HIMSI</i>	HIMSI is sent from UE to eNB2(T) securely
<i>secrecy_of_ue_enbt_UECap</i>	UECap is sent from UE to eNB2(T) securely
<i>secrecy_of_enbt_enbs_RNe2</i>	RNe2 is sent from eNB2(T) to eNB1(S) securely
<i>secrecy_of_enbt_enbs_eID2</i>	eID2 is sent from eNB2(T) to eNB1(S) securely
<i>secrecy_of_enbt_enbs_HIMSI</i>	HIMSI is sent from eNB2(T) to eNB1(S) securely
<i>secrecy_of_enbt_enbs_e1ueAuthN</i>	e1ueAuthN is sent from eNB2(T) to eNB1(S) securely
<i>secrecy_of_enbt_ue_Xe1ueAuthN</i>	Xe1ueAuthN is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_RNe2</i>	RNe2 is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_RNe1</i>	RNe1 is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_mme_UECap</i>	UECap is sent from eNB2(T) to MME securely
<i>secrecy_of_enbt_mme_RNe2N</i>	RNe2N is sent from eNB2(T) to MME securely
<i>secrecy_of_enbt_mme_HIMSI</i>	HIMSI is sent from eNB2(T) to MME securely
<i>secrecy_of_enbt_mme_MVP</i>	MVP is sent from eNB2(T) to MME securely
<i>secrecy_of_enbs_enbt_HIMSI</i>	HIMSI is sent from eNB1(S) to eNB2(T) securely
<i>secrecy_of_enbt_ue_RNueN</i>	RNueN is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_Xe2ueAuth</i>	Xe2ueAuth is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_RNue</i>	RNue is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_FSKe2u</i>	FSKe2u is sent from eNB2(T) to UE securely
<i>secrecy_of_enbt_ue_RNe2N</i>	RNe2N is sent from eNB2(T) to UE securely
<i>secrecy_of_enbs_enbt_KeNBN</i>	KeNBN is sent from eNB1(S) to eNB2(T) securely
<i>secrecy_of_enbs_enbt_Xe1ueAuthN</i>	Xe1ueAuthN is sent from eNB1(S) to eNB2(T) securely
<i>secrecy_of_enbs_mme_HIMSI</i>	HIMSI is sent from eNB1(S) to MME securely

The results of REPS-AKA2 protocol in OFMC and CL-Atse modes are shown in Figure 19 and Figure 20 respectively (Details are in Appendix1). Here, “SAFE” is the keyword showing that the proposed protocol is safe.

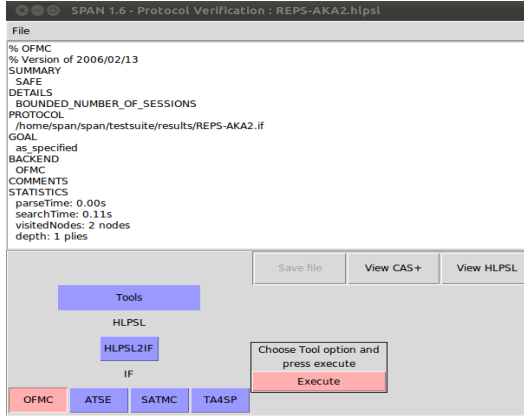


Figure 19: Evaluation result in OFMC

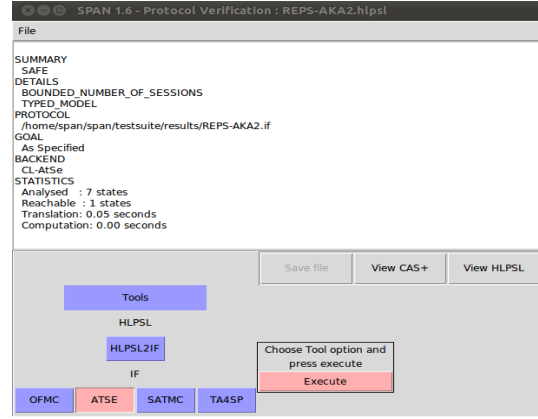


Figure 20: Evaluation result in CL-AtSe

4.2.2 Formal analysis using Scyther

The suggested REPS-AKA2 method is evaluated by using Scyther tool. The description of the presented REPS-AKA2 protocol is implemented based on the defined nodes. There are four nodes (UE, eNodeB1, eNodeB2, and MME) in REPS-AKA2 protocol. In the modeled protocol, five claims (i.e., Nisynch, Secrecy, Weakagree, Niagree, and Alive) are checked for mentioned nodes. (Scyther codes are presented in Appendix 2).

Based on UE's viewpoint in REPS-AKA2, the security claims are presented as follows:
 $claim_ue1(UE, Secret, HMSI); claim_ue2(UE, Secret, UECap); claim_ue3(UE, Secret, RNue); claim_ue4(UE, Secret, eID1); claim_ue5(UE, Secret, e1ueAuthP); claim_ue6(UE, Secret, RNe2); claim_ue7(UE, Secret, e2ueAuth); claim_ue8(UE, Secret, RNueP); claim_ue9(UE, Secret, UECapP); claim_ue10(UE, Secret, T1P); claim_ue11(UE, Secret, e2ueAuthP); claim_uea(UE, Alive); claim_uew(UE, Weakagree); claim_uenia(UE, Niagree); claim_uenis(UE, Nisynch);$

Based on eNodeB1's viewpoint in REPS-AKA2, the security claims are presented as follows:
 $claim_enb1c1(ENB1, Secret, HMSI); claim_enb1c2(ENB1, Secret, KeNB1P); claim_enb1c3(ENB1, Secret, RNe2); claim_enb1c4(ENB1, Secret, RNe1); claim_enb1c5(ENB1, Secret, RNue); claim_enb1c6(ENB1, Secret, T3); claim_enb1c7(ENB1, Secret, X_{e1ue}AuthP); claim_enb1c8(ENB1, Secret, UECap); claim_enb1c9(ENB1, Secret, eID2); claim_enb1c10(ENB1, Secret, NKG); claim_enb1c11(ENB1, Secret, MVP); claim_enb1a(ENB1, Alive); claim_enb1w(ENB1, Weakagree); claim_enb1nia(ENB1, Niagree); claim_enb1nis(ENB1, Nisynch);$

Based on eNodeB2's viewpoint in REPS-AKA2, the security claims are presented as follows:
 $claim_enb2c1(ENB2, Secret, UECap); claim_enb2c2(ENB2, Secret, RNe2); claim_enb2c3(ENB2, Secret, T2); claim_enb2c4(ENB2, Secret, RNue); claim_enb2c5(ENB2, Secret, eID2); claim_enb2c6(ENB2, Secret, HMSI); claim_enb2c7(ENB2, Secret, X_{e1ue}AuthP); claim_enb2c8(ENB2, Secret, RNe2); claim_enb2c9(ENB2, Secret, RNe1); claim_enb2c10(ENB2, Secret, RNue); claim_enb2c11(ENB2, Secret, T4); claim_enb2c12(ENB2, Secret, FS_{Ke2u}); claim_enb2c13(ENB2, Secret, X_{e1ue}AuthP); claim_enb2c14(ENB2, Secret, RNe2P); claim_enb2c15(ENB2, Secret, RNueP); claim_enb2c16(ENB2, Secret, T2P); claim_enb2c17(ENB2, Secret, X_{e2ue}Auth); claim_enb2c18(ENB2, Secret, UECapP); claim_enb2c19(ENB2, Secret, UK); claim_enb2c20(ENB2, Secret, MVP); claim_enb2a(ENB2, Alive); claim_enb2w(ENB2, Weakagree); claim_enb2nia(ENB2, Niagree); claim_enb2nis(ENB2, Nisynch);$

Based on MME's viewpoint in REPS-AKA2, the security claims are presented as follows:
 $claim_mmea(MME, Alive); claim_mmew(MME, Weakagree); claim_mmenia(MME, Niagree); claim_mmenis(MME, Nisynch);$

The evaluation results of modeled protocol are displayed in Figure 21-24. As it can be seen, the proposed protocols satisfy all defined security claims in involved nodes (UE, MME, eNodeB1 and eNodeB2).

Claim				Status		Comments
REPSAKA2	UE	REPSAKA2,ue1	Secret H1(IMS1)	Ok	Verified	No attacks.
		REPSAKA2,ue2	Secret UECap	Ok	Verified	No attacks.
		REPSAKA2,ue3	Secret RNue	Ok	Verified	No attacks.
		REPSAKA2,ue4	Secret eID1	Ok	Verified	No attacks.
		REPSAKA2,ue5	Secret AuthFunc(XOR(e1ueAuth,RNue))	Ok	Verified	No attacks.
		REPSAKA2,ue6	Secret RNe2	Ok	Verified	No attacks.
		REPSAKA2,ue7	Secret AuthFunc(XOR(RNe1,RNe2))	Ok	Verified	No attacks.
		REPSAKA2,ue8	Secret RNueP	Ok	Verified	No attacks.
		REPSAKA2,ue9	Secret UECapP	Ok	Verified	No attacks.
		REPSAKA2,ue10	Secret T1P	Ok	Verified	No attacks.
		REPSAKA2,ue11	Secret AuthFunc(XOR(AuthFunc(XOR(RNe1,RNe2)),RNueP...	Ok	Verified	No attacks.
		REPSAKA2,uea	Alive	Ok	Verified	No attacks.
		REPSAKA2,uew	Weakagree	Ok	Verified	No attacks.
		REPSAKA2,uenia	Niagree	Ok	Verified	No attacks.
		REPSAKA2,uenis	Nisynch	Ok	Verified	No attacks.

Figure 21: Scyther evaluation for defined claims in the UE

Claim				Status		Comments
REPSAKA2	ENB2	REPSAKA2,enb2c2	Secret RNe2	Ok	Verified	No attacks.
		REPSAKA2,enb2c4	Secret RNue	Ok	Verified	No attacks.
		REPSAKA2,enb2c6	Secret H1(IMS1)	Ok	Verified	No attacks.
		REPSAKA2,enb2c7	Secret AuthFunc(XOR(e1ueAuth,RNue))	Ok	Verified	No attacks.
		REPSAKA2,enb2c9	Secret RNe1	Ok	Verified	No attacks.
		REPSAKA2,enb2c12	Secret FSKGFunc(RNe2,H1(IMS1))	Ok	Verified	No attacks.
		REPSAKA2,enb2c13	Secret KDF(XOR(e1ueAuth,RNe1))	Ok	Verified	No attacks.
		REPSAKA2,enb2c14	Secret RNe2P	Ok	Verified	No attacks.
		REPSAKA2,enb2c15	Secret RNueP	Ok	Verified	No attacks.
		REPSAKA2,enb2c17	Secret AuthFunc(XOR(AuthFunc(XOR(RNe1,RNe2)),RNe2P...	Ok	Verified	No attacks.
		REPSAKA2,enb2c18	Secret UECapP	Ok	Verified	No attacks.
		REPSAKA2,enb2c19	Secret UKGFunc(RNe2P,RNueP,AuthFunc(XOR(RNe1,RNe2)...	Ok	Verified	No attacks.
		REPSAKA2,enb2c20	Secret MVP	Ok	Verified	No attacks.
		REPSAKA2,enb2a	Alive	Ok	Verified	No attacks.
		REPSAKA2,enb2w	Weakagree	Ok	Verified	No attacks.
		REPSAKA2,enb2nia	Niagree	Ok	Verified	No attacks.
		REPSAKA2,enb2nis	Nisynch	Ok	Verified	No attacks.

Figure 22: Scyther evaluation for defined claims in the eNodeB2

Claim				Status		Comments
REPSAKA2	MME	REPSAKA2,mmea	Alive	Ok	Verified	No attacks.
		REPSAKA2,mmew	Weakagree	Ok	Verified	No attacks.
		REPSAKA2,mmenia	Niagree	Ok	Verified	No attacks.
		REPSAKA2,mmenis	Nisynch	Ok	Verified	No attacks.

Figure 23: Scyther evaluation for defined claims in the MME

Claim				Status		Comments
REPSAKA2	ENB1	REPSAKA2,enb1c1	Secret H1(IMSI)	Ok	Verified	No attacks.
		REPSAKA2,enb1c2	Secret KDF(KeNB,NKGFunc(RNe2,RNue,RNe1))	Ok	Verified	No attacks.
		REPSAKA2,enb1c3	Secret RNe2	Ok	Verified	No attacks.
		REPSAKA2,enb1c4	Secret RNe1	Ok	Verified	No attacks.
		REPSAKA2,enb1c5	Secret RNue	Ok	Verified	No attacks.
		REPSAKA2,enb1c6	Secret T3	Ok	Verified	No attacks.
		REPSAKA2,enb1c7	Secret KDF(XOR(e1ueAuth,RNe1))	Ok	Verified	No attacks.
		REPSAKA2,enb1c8	Secret UECap	Ok	Verified	No attacks.
		REPSAKA2,enb1c9	Secret eID2	Ok	Verified	No attacks.
		REPSAKA2,enb1c10	Secret NKGFunc(RNe2,RNue,RNe1)	Ok	Verified	No attacks.
		REPSAKA2,enb1c11	Secret MVP	Ok	Verified	No attacks.
		REPSAKA2,enb1a	Alive	Ok	Verified	No attacks.
		REPSAKA2,enb1w	Weakagree	Ok	Verified	No attacks.
		REPSAKA2,enb1nia	Niagree	Ok	Verified	No attacks.
		REPSAKA2,enb1nis	Nisynch	Ok	Verified	No attacks.

Figure 24: Scyther evaluation for defined claims in the eNodeB1

4.2.3 Security analysis by BAN logic

In this section, we prove that AKA procedure of the REPS-AKA2 method is safe by means of BAN Logic. This logic is used to perform a security verification of different AKA protocols. We verify the transmitted messages to check confidentiality among involved nodes in the REPS-AKA2 method.

The formal messages of the proposed protocol REPS-AKA2 are as follows: **M1**: $UE \rightarrow eNB2$: $\{HIMSI, UECap, RNue, eID1, PKue, \{e1ueAuth^*\}_{inv(PKue)}\}_{PKe2}$, **M2**: $eNB2 \rightarrow eNB1$: $\{UECap, RNe2, T2, RNue, eID2, \{HIMSI\}_{inv(PKe2)}, \{e1ueAuth^*\}_{inv(PKue)}\}_{PKe1}$, **M3**: $eNB1 \rightarrow eNB2$: $\{HIMSI, KeNB^*, RNe2, RNe1, RNue, T3, \{Xe1ueAuth^*\}_{inv(PKe1)}\}_{PKe2}$, **M4**: $eNB1 \rightarrow MME$: $\{HIMSI, UECap, eID2, NKG, T3, RNe1, \{MVP\}_{inv(PKe1)}\}_{PKm}$, **M5**: $eNB2 \rightarrow UE$: $\{RNe2, RNe1, RNue, T4, \{FSKe2u\}_{inv(PKe2)}, \{Xe1ueAuth^*\}_{inv(PKe1)}\}_{PKue}$, **M6**: $UE \rightarrow eNB2$: $\{RNe2, \{e2ueAuth^*\}_{inv(PKue)}\}_{FSKe2u}$, **M1'**: $UE \rightarrow eNB2$: $\{HIMSI, RNue', UECap', eID1, T1', \{e2ueAuth^*\}_{inv(PKue)}\}_{FSKe2u}$, **M2'**: $eNB2 \rightarrow UE$: $\{RNe2', RNue', T2', \{Xe2ueAuth\}_{inv(PKe2)}\}_{FSKe2u}$, **M3'**: $eNB2 \rightarrow MME$: $\{UECap', RNe2', T2', UK, HIMSI, \{MVP\}_{inv(PKe2)}\}_{PKm}$.

We need to transform all messages into idealized logical format as follows: **M1**: $UE \rightarrow eNB2$: $eNB2 \triangleleft \{UE \mid \Rightarrow HIMSI, \#(UECap), \#(RNue), \#(eID1), \xrightarrow{PK_{UE}} UE, \{UE \mid \Rightarrow e1ueAuth^*\}_{PKue^{-1}}\}_{PKe2}$, **M2**: $eNB2 \rightarrow eNB1$: $eNB1 \triangleleft \{UE \mid \Rightarrow UECap, \#(RNe2), \#(T2), UE \mid \Rightarrow RNue, \#(eID2), \{UE \mid \Rightarrow HIMSI\}_{PKe2^{-1}}, \{UE \mid \Rightarrow e1ueAuth^*\}_{PKue^{-1}}\}_{PKe1}$, **M3**: $eNB1 \rightarrow eNB2$: $eNB2 \triangleleft \{UE \mid \Rightarrow HIMSI, eNB1 \mid \Rightarrow KeNB^*, eNB2 \mid \Rightarrow RNe2, \#(RNe1), UE \mid \Rightarrow RNue, \#(T3), \{eNB1 \mid \Rightarrow Xe1ueAuth^*\}_{PKe1^{-1}}\}_{PKe2}$, **M4**: $eNB1 \rightarrow MME$: $MME \triangleleft \{UE \mid \Rightarrow \{HIMSI, UECap, RNe1\}, eNB2 \mid \Rightarrow eID2, eNB1 \mid \Rightarrow NKG, \#(T3), \{eNB1 \mid \Rightarrow MVP\}_{PKe1^{-1}}\}_{PKm}$, **M5**: $eNB2 \rightarrow UE$: $UE \triangleleft \{eNB2 \mid \Rightarrow RNe2, eNB1 \mid \Rightarrow RNe1, UE \mid \Rightarrow RNue, \#(T4), \{eNB2 \mid \Rightarrow FSKe2u\}_{PKe2^{-1}}, \{eNB1 \mid \Rightarrow Xe1ueAuth^*\}_{PKe1^{-1}}\}_{PKue}$, **M6**: $UE \rightarrow eNB2$: $eNB2 \triangleleft \{eNB2 \mid \Rightarrow RNe2, \{UE \mid \Rightarrow e2ueAuth\}_{PKue^{-1}}\}_{FSKe2u}$, **M1'**: $UE \rightarrow eNB2$: $eNB2 \triangleleft \{UE \mid \Rightarrow HIMSI, \#(RNue'), \#(UECap'), \#(eID1), \#(T1'), \{UE \mid \Rightarrow e2ueAuth^*\}_{PKue^{-1}}\}_{FSKe2u}$, **M2'**: $eNB2 \rightarrow UE$: $UE \triangleleft \{\#(RNe2'), UE \mid \Rightarrow RNue', \#(T2'), \{eNB2 \mid \Rightarrow Xe2ueAuth\}_{PKe2^{-1}}\}_{FSKe2u}$, **M3'**: $eNB2 \rightarrow MME$: $MME \triangleleft \{UE \mid \Rightarrow \{UECap', HIMSI\}, \#(RNe2'), \#(T2'), eNB2 \mid \Rightarrow UK, \{eNB2 \mid \Rightarrow MVP\}_{PKe2^{-1}}\}_{PKm}$.

The following initial Assumptions (A) are considered to evaluate the proposed protocol: **A1**: It is considered that some public keys in REPS-AKA2 (i.e. PKue, PKe1, PKe2 and PKm) are used between involved entities as follows: $UE \models \xrightarrow{PK_{ue}} UE \wedge \xrightarrow{PK_{e1}} eNB1$

$\wedge \xrightarrow{PK_{e2}} eNB2, MME \models \xrightarrow{PK_m} MME \wedge \xrightarrow{PK_{e1}} eNB1, eNB1 \models \xrightarrow{PK_{ue}} UE \wedge \xrightarrow{PK_{e1}} eNB1 \wedge \xrightarrow{PK_{e2}} eNB2 \wedge$

$\xrightarrow{PK_m} MME, eNB2 \models \xrightarrow{PK_{ue}} UE \wedge \xrightarrow{PK_{e1}} eNB1 \wedge \xrightarrow{PK_{e2}} eNB2$, **A2:** It is considered that some secret keys in REPS-AKA2 (i.e. FSK_{e2u}) are used between different nodes as follows: $UE \models UE \xleftrightarrow{FSK_{e2u}} eNB2, eNB2 \models UE \xleftrightarrow{FSK_{e2u}} eNB2$, **A3:** It is assumed that the entities calculate (i.e., control) some security parameters and generated keys as follows: $UE \models HIMS1 \wedge NKG \wedge KeNB^* \wedge KRRCint \wedge KRRCenc \wedge KRRUPenc \wedge KNASenc \wedge KNASint \wedge KASME^* \wedge e1ueAuth^* \wedge e2ueAuth \wedge e2ueAuth^* \wedge Xe2ueAuth \wedge UK \wedge e2ueAuth^{**} \wedge KRRCint^* \wedge KRRCenc^* \wedge KRRUPenc^* \wedge KNASenc^* \wedge KNASint^*$, $MME \models HNID \wedge KASME^* \wedge KNASenc \wedge KNASint \wedge KNASenc^* \wedge KNASint^*$, $eNB1 \models NKG \wedge KeNB^* \wedge Xe1ueAuth^*$, $eNB2 \models KRRCint \wedge KRRCenc \wedge KRRUPenc \wedge FSK_{e2u} \wedge e2ueAuth \wedge e2ueAuth^* \wedge e2ueAuth^{**} \wedge UK \wedge KRRCint^* \wedge KRRCenc^* \wedge KRRUPenc^* \wedge Xe2ueAuth$, **A4:** We assumed that the entities generate some random value to use in key derivation and authentication functions as followa: $UE \models \#(RNue)$, $eNB1 \models \#(RNe1)$, $eNB2 \models \#(RNe2)$, **A5:** It is considered that the entities have some security value related to other involved nodes as follows: $UE \models (UE \xleftrightarrow{NAS-CAD} MME) \wedge (UE \xleftrightarrow{NAS-IAD} MME) \wedge (UE \xleftrightarrow{AS-IAD} eNB2) \wedge (UE \xleftrightarrow{AS-EAD} eNB2) \wedge (UE \xleftrightarrow{AS-UPEAD} eNB2) \wedge (UE \xleftrightarrow{UECap} eNB1) \wedge (UE \xleftrightarrow{UECap} eNB2) \wedge (UE \xleftrightarrow{UECap} MME)$, $MME \models (UE \xleftrightarrow{NAS-CAD} MME) \wedge (UE \xleftrightarrow{NAS-IAD} MME) \wedge (UE \xleftrightarrow{UECap} MME) \wedge (eNB1 \xleftrightarrow{MVP} MME)$, $eNB1 \models (eNB1 \xleftrightarrow{eID1} eNB2) \wedge (eNB1 \xleftrightarrow{eID2} eNB2) \wedge (UE \xleftrightarrow{UECap} eNB1) \wedge (eNB1 \xleftrightarrow{MVP} MME)$, $eNB2 \models (UE \xleftrightarrow{UECap} eNB1) \wedge (eNB2 \xleftrightarrow{UECap} eNB1) \wedge (eNB1 \xleftrightarrow{eID1} eNB2) \wedge (eNB1 \xleftrightarrow{eID2} eNB2) \wedge (UE \xleftrightarrow{AS-IAD} eNB2) \wedge (UE \xleftrightarrow{AS-EAD} eNB2) \wedge (UE \xleftrightarrow{AS-UPEAD} eNB2)$.

The main security Goals (G) of the suggested protocol (i.e., REPS-AKA2) are to provide the mutual authentication between involved nodes and manage security keys between involved entities. All defined security goals in the suggested method are represented as follows: **G1:** The mutual authentication between involved nodes is ensured in the suggested approaches as $UE \models eNB1 \models eNB2 \wedge eNB2 \models eNB1 \models UE$ then $UE \models eNB2 \wedge eNB2 \models UE$, **G2:** Key acceptance achieved between different defined entities. There is a matched KASME between the UE, HSS, and MME to run the proper key

agreement. As $MME \models UE \models KASME \wedge UE \models MME \models KASME$ then $UE \xleftrightarrow{KASME} MME$, thus, as $UE \xleftrightarrow{KASME} HSS$ then $UE \xleftrightarrow{KNASenc} MME \wedge UE \xleftrightarrow{KNASint} MME$. Also, there is a matched $KeNB^*$ between the UE, and eNB2 to run the key agreement. As $eNB2 \models UE \models KeNB^* \wedge UE \models eNB2 \models KeNB^*$ then $UE \xleftrightarrow{KeNB^*} eNB2$, thus, as $UE \xleftrightarrow{KeNB^*} eNB2$ then $UE \xleftrightarrow{KRRCint^*} eNB2 \wedge UE \xleftrightarrow{KRRCenc^*} eNB2 \wedge UE \xleftrightarrow{KRRUPenc^*} eNB2$, **G3:** Message authentication between corresponding entities as follows: $eNB2 \models UE \models e1ueAuth^*$ (M1), $eNB1 \models eNB2 \models \sim HIMS1$ (M2), $eNB1 \models UE \models \sim e1ueAuth^*$ (M2), $MME \models eNB1 \models \sim MVP$ (M4), $eNB2 \models eNB1 \models \sim Xe1ueAuth^*$ (M3), $UE \models eNB2 \models \sim FSKe2u$ (M5), $UE \models eNB1 \models \sim Xe1ueAuth^*$ (M5), $eNB2 \models UE \models \sim e2ueAuth$ (M6), $eNB2 \models UE \models \sim e2ueAuth^*$ (M1'), $UE \models eNB2 \models \sim Xe2ueAuth$ (M2'), $MME \models eNB2 \models \sim MVP$ (M3'), **G4:** Key agreement in AS and NAS layers are satisfied in involved nodes as follows: $UE \models HSS \models (X) RNK \wedge HSS \models UE \models (X) RNK$ then $UE \xleftrightarrow{(X) RNK} HSS$ and also $UE \models eNB1 \models NKG \wedge eNB1 \models UE \models NKG$ then $UE \xleftrightarrow{NKG} eNB1$, and also $UE \models eNB2 \models UK \wedge eNB2 \models UE \models UK$ then $UE \xleftrightarrow{UK} eNB2$, thus these involved nodes can generate some agreed $KeNB$, UK , and $KASME$, **G5:** The key freshness between UE and network side (i.e., MME, eNB1, and eNB2) and also among network nodes. The key freshness between these entities holds since $UE \models eNB1 \models \#(RNue) \wedge \#(RNe1) \wedge \#(RNe2)$, $eNB1 \models UE \models \#(RNue) \wedge \#(RNe1) \wedge \#(RNe2)$, $\{UE, eNB1\} \models eNB2 \models \#(RNue) \wedge \#(RNe1) \wedge \#(RNe2)$, thus, $KASME$, NKG , $KeNB^*$, and $KASME^*$ are fresh keys accordingly, **G6:** Confidentiality between the UE (as a user side) and the eNB2 (as a network side):
$$\frac{eNB2 \models (eNB2 \xleftrightarrow{KRRCenc} UE), UE \triangleleft \{Z\}_{KRRCenc}}{eNB2 \models UE \models \{Z\}_{KRRCenc}} \wedge \frac{UE \models (eNB2 \xleftrightarrow{KRRCenc} UE), eNB2 \triangleleft \{R\}_{KRRCenc}}{UE \models eNB2 \models \{R\}_{KRRCenc}}$$
, Where 'Z' and 'R' are transmitted messages from UE and eNB2 respectively.

We evaluate the idealized form of presented method according to the hypothesis we defined and the BAN Logic rules including Message meaning rule (R1), Jurisdiction

rule (R2), and Nonce Verification rule (R3). We employ the BAN logic rule (R2) to

derive: $\frac{UE \mid \equiv eNB1 \Rightarrow e1ueAuth^*, UE \mid \equiv eNB1 \mid \equiv e1ueAuth^*}{UE \mid \equiv e1ueAuth^*}, \frac{eNB1 \mid \equiv UE \Rightarrow e1ueAuth^*, eNB1 \mid \equiv UE \mid \equiv e1ueAuth^*}{eNB1 \mid \equiv e1ueAuth^*}$ To

satisfy G1. We utilize the BAN logic rule (R2) to derive:

$\frac{UE \mid \equiv eNB2 \Rightarrow KeNB^*, UE \mid \equiv eNB2 \mid \equiv KeNB^*}{UE \mid \equiv KeNB^*}, \frac{eNB2 \mid \equiv UE \Rightarrow KeNB^*, eNB2 \mid \equiv UE \mid \equiv KeNB^*}{eNB2 \mid \equiv KeNB^*}$, To satisfy G2. We use

the BAN logic rule (R1) to derive:

$\frac{eNB2 \mid \equiv \left(\frac{PK_{ue}}{\rightarrow} UE \right), eNB2 \triangleleft \{e1ueAuth^*\}_{PK_{ue}^{-1}}}{eNB2 \mid \equiv UE \mid \sim e1ueAuth^*}, \frac{eNB1 \mid \equiv \left(\frac{PK_{e2}}{\rightarrow} eNB2 \right), eNB1 \triangleleft \{HIMSI\}_{PK_{e2}^{-1}}}{eNB1 \mid \equiv eNB2 \mid \sim HIMSI},$

$\frac{eNB1 \mid \equiv \left(\frac{PK_{ue}}{\rightarrow} UE \right), eNB1 \triangleleft \{e1ueAuth^*\}_{PK_{ue}^{-1}}}{eNB1 \mid \equiv UE \mid \sim e1ueAuth^*}, \frac{MME \mid \equiv \left(\frac{PK_{e1}}{\rightarrow} eNB1 \right), MME \triangleleft \{MVP\}_{PK_{e1}^{-1}}}{MME \mid \equiv eNB1 \mid \sim MVP},$

$\frac{eNB2 \mid \equiv \left(\frac{PK_{e1}}{\rightarrow} eNB1 \right), eNB2 \triangleleft \{Xe1ueAuth^*\}_{PK_{e1}^{-1}}}{eNB2 \mid \equiv eNB1 \mid \sim Xe1ueAuth^*}, \frac{UE \mid \equiv \left(\frac{PK_{e2}}{\rightarrow} eNB2 \right), UE \triangleleft \{FSKe2u\}_{PK_{e2}^{-1}}}{UE \mid \equiv eNB2 \mid \sim FSKe2u},$

$\frac{UE \mid \equiv \left(\frac{PK_{e1}}{\rightarrow} eNB1 \right), UE \triangleleft \{Xe1ueAuth^*\}_{PK_{e1}^{-1}}}{UE \mid \equiv eNB1 \mid \sim Xe1ueAuth^*}, \frac{eNB2 \mid \equiv \left(\frac{PK_{ue}}{\rightarrow} UE \right), eNB2 \triangleleft \{e2ueAuth\}_{PK_{ue}^{-1}}}{eNB2 \mid \equiv UE \mid \sim e2ueAuth},$

$\frac{UE \mid \equiv \left(\frac{PK_{e2}}{\rightarrow} eNB2 \right), UE \triangleleft \{Xe2ueAuth^*\}_{PK_{e2}^{-1}}}{UE \mid \equiv eNB2 \mid \sim Xe2ueAuth^*}, \frac{eNB2 \mid \equiv \left(\frac{PK_{ue}}{\rightarrow} UE \right), eNB2 \triangleleft \{e2ueAuth^*\}_{PK_{ue}^{-1}}}{eNB2 \mid \equiv UE \mid \sim e2ueAuth^*},$

$\frac{MME \mid \equiv \left(\frac{PK_{e2}}{\rightarrow} eNB2 \right), MME \triangleleft \{MVP\}_{PK_{e2}^{-1}}}{MME \mid \equiv eNB2 \mid \sim MVP}$, To satisfy G3. We employ the BAN logic rule (R2) to

derive: $\frac{eNB1 \mid \equiv UE \Rightarrow NKG, eNB1 \mid \equiv UE \mid \equiv NKG}{eNB1 \mid \equiv NKG}, \frac{UE \mid \equiv eNB1 \Rightarrow NKG, UE \mid \equiv eNB1 \mid \equiv NKG}{UE \mid \equiv NKG},$

$\frac{eNB2 \mid \equiv UE \Rightarrow UK, eNB2 \mid \equiv UE \mid \equiv UK}{eNB2 \mid \equiv UK}, \frac{UE \mid \equiv eNB2 \Rightarrow UK, UE \mid \equiv eNB2 \mid \equiv UK}{UE \mid \equiv UK}$ To satisfy G4. And then all

derived keys and parameters from NKG, UK and (X) RNK can be proved accordingly.

We employ the BAN logic rule (R3) to derive:

$\frac{UE \mid \equiv \#(RNue), UE \mid \equiv eNB2 \mid \sim RNue}{UE \mid \equiv eNB2 \mid \equiv RNue} \wedge \frac{UE \mid \equiv eNB2 \mid \Rightarrow RNue, UE \mid \equiv eNB2 \mid \equiv RNue}{UE \mid \equiv RNue}$

$\frac{eNB2 \mid \equiv \#(RNe2), eNB2 \mid \equiv UE \mid \sim RNe2}{eNB2 \mid \equiv UE \mid \equiv RNe2} \wedge \frac{eNB2 \mid \equiv UE \mid \Rightarrow RNe2, eNB2 \mid \equiv UE \mid \equiv RNe2}{eNB2 \mid \equiv RNe2}$ Thus, all derived keys (i.e.,

KRRCint, KRRCenc, and KRRUPenc) are generated using fresh numbers. Hence the

protocol satisfies G5. Using proofs in G4, we learn that $UE \mid \equiv eNB2 \mid \equiv KeNB \wedge eNB2 \mid \equiv$

$UE \mid \equiv KeNB$. And by G5 and (R3): $\frac{UE \mid \equiv \#(RNe2), UE \mid \equiv eNB2 \mid \sim RNe2}{UE \mid \equiv RNe2} \wedge$

$\frac{eNB2 \mid \equiv \#(RNue), eNB2 \mid \equiv UE \mid \sim RNue}{eNB2 \mid \equiv RNue}$ Thus, we learn NKG is fresh and depend on RNue and

RNe2 which are shared between entities, thus we have $UE \xleftrightarrow{KRRCenc} eNB2$ to satisfy G6.

4.2.4 Proof of security requirements

In this section, correctness proofs of the REPS-AKA2 are illustrated as follows:

Claim 1: The REPS-AKA2 can optimize the computational cost of UE.

Proof: During proposed handover in REPS-AKA2, the user node (i.e., UE) configure the UECap parameter. The UECap specify encryption capability of each UE node. As different UEs have different hardware/software specification, they might not support all encryption algorithms properly. This point can affect the level of QoS in handover and key agreement process. According to the UECap, the UE and LTE network set AS-IAD, AS-EAD, AS-UPEAD, NAS-CAD, and NAS-IAD values. These parameters are used to generate KRRCCint, KRRCCenc, KRRCCUPenc, KNASenc and KNASint.

Claim 2: The protocol is resistant to Replay attacks in X2-handover.

Proof: We use timestamps in our method to protect the system against replay attack. The involved nodes verify the timestamps to confirm that the messages are fresh. In the protocol, stale messages are ignored by eNodeBs. Thus, these dropped messages cannot be forwarded to the MME. Each node has timestamps and fresh random number. These pair parameters (e.g. (T1, RNue)) provide freshness of messages. Thus, an attacker cannot use the same message frequently, decreasing the replay attacks.

Claim 3: The REPS-AKA2 achieves protection of UE's Identity in X2-handover.

Proof: The UE does not forward IMSI values in a clear text. Since the IMSI parameter, the system should hide it from the attacker. Consequently, we hash IMSI (HIMSI=H1(IMSI)) and then transmit HISMI encrypted by eNodeB public key (PKe2).

Claim 4: The REPS-AKA2 achieves backward key separation.

Proof: To satisfy backward key separation in REPS-AKA2, source eNodeB produces the next K_{eNB} by using a one-way hash function and NKG, a new key generator. The NKG is generated based on random values, source eNodeB, and target eNodeB; $NKG = NKG_{Func}(R_{Ne2}, R_{Nue}, R_{Ne1})$. In this scheme, NKG is always a fresh value, and it is generated for only one handover between target eNodeB and source eNodeB.

Claim 5: The REPS-AKA2 provides authentication between UE and eNodeBs.

Proof: The UE and source eNodeB can authenticate each other using $e1ueAuth^*$ and $Xe1ueAuth^*$. The UE will generate the $e1ueAuth^*$ by $e1ueAuth$ (from last handover) and R_{Nue} ; $e1ueAuth^* = Auth_{Func}(e1ueAuth \oplus R_{Nue})$ and transmit through source eNodeB. The source eNodeB can make the same $e1ueAuth^*$ because it has $e1ueAuth$ from the last handover and received R_{Nue} and match with received $e1ueAuth^*$ in M2. On the other hand, the source eNodeB will perform the same procedure on the other side and make $Xe1ueAuth^*$ by $e1ueAuth$ from the previous handover and R_{Ne1} (a fresh random number). Consequently, the UE can generate same $Xe1ueAuth^*$ by received R_{Ne1} and compare with received $Xe1ueAuth^*$ from M5; $Xe1ueAuth^* = Auth_{Func}(e1ueAuth \oplus R_{Ne1})$. Thus, UE and source eNodeB can authenticate each other considerably, and the attacker cannot send any message to source eNodeB and run his DoS attacks easily. The suggested approach provides robust mutual authentication between UE and target eNodeB. They check each other using R_{Nue} and R_{Ne2} . Target eNodeB and source eNodeB authenticate each other using R_{Ne2} and R_{Nue} . Thus, the REPS-AKA2 can protect the LTE system against

MitM attacks. The rogue eNodeB cannot impersonate itself to UE and source eNodeB. The reason is that the $e1ueAuth$ value is saved in the UE and valid source eNodeB.

Claim 6: The REPS-AKA2 can provide message authentication in X2-handover.

Proof: The target eNodeB can authentication M1 by $\{e1ueAuth^*\}_{inv(PKue)}$. The $e1ueAuth^*$ value is authenticated by target eNodeB as produced by a valid UE. Details of message authentication in other messages are shown in Table 9.

Table 9: Description of message authentication in REPS-AKA2

Messages	Authentication Descriptions
<i>M1 (UE $\rightarrow$ eNB2)</i>	The $e1ueAuth^*$ is verified by eNB2 as generated by UE
<i>M2 (eNB2 $\rightarrow$ eNB1)</i>	The HIMS1 is verified by eNB1 as generated by eNB2
<i>M3 (eNB1 $\rightarrow$ eNB2)</i>	The $Xe1ueAuth^*$ is authenticated by eNB2 as built by eNB1
<i>M4 (eNB1 $\rightarrow$ MME)</i>	The MVP is verified by MME as generated by eNB1
<i>M5 (eNB2 $\rightarrow$ UE)</i>	The FSK _{e2u} is verified by UE as generated by eNB2
<i>M6 (UE $\rightarrow$ eNB2)</i>	The $e2ueAuth$ is verified by eNB2 as generated by UE
<i>M1' (UE $\rightarrow$ eNB2)</i>	The $e2ueAuth^*$ is verified by eNB2 as generated by UE
<i>M2' (eNB2 $\rightarrow$ UE)</i>	The $Xe2ueAuth$ is verified by UE as generated by eNB2
<i>M3' (eNB2 $\rightarrow$ MME)</i>	The MVP is verified by MME as generated by eNB2

Claim 7: The protocol can provide robust key agreement in AS layer.

Proof: The keys of RRC layer in the UE and LTE network (target eNodeB) are calculated using new $KeNB^*$ and XOR of $RNue$ and $RNe2$; $KRRCint = KDF(KeNB^*, (RNue \oplus RNe2), AS-IAD)$, $KRRCenc = KDF(KeNB^*, (RNue \oplus RNe2), AS-EAD)$, $KRRCUPenc = KDF(KeNB^*, (RNue \oplus RNe2), AS-UPEAD)$. Thus, they are fresh and based on new random values (i.e., $RNue$ and $RNe2$). The attackers are unable to extract them for the future handover. Thus, the security goals in AS layer can be satisfied.

Claim 8: The REPS-AKA2 can provide strong key agreement in NAS layer.

Proof: The KNASint, and KNASenc keys are generated by using calculated KASME* key. Since the KASME* is generated based on KASME and the new key generator (i.e., NKG), KNASenc and KNASint have a suitable security and the ability to make NAS keys securely.

Claim 9: The protocol achieves prevention of eavesdropping attacks.

Proof: During X2-handover in existing EPS-AKA method, adversary can extract the transmitted messages in communication networks. However, in our method, adversaries are unable to monitor transmitted messages. Since all forwarded messages are encrypted by Symmetric/Asymmetric keys as presented in Table 10. Thus, a suitable confidentiality is provided properly in all transmitted messages.

Table 10: Asymmetric and symmetric keys in REPS-AKA2

Messages	Keys	Encryption types
$M1 (UE \rightarrow eNB2)$	PKe2	Asymmetric
$M2 (eNB2 \rightarrow eNB1)$	PKe1	Asymmetric
$M3 (eNB1 \rightarrow eNB2)$	PKe2	Asymmetric
$M4 (eNB1 \rightarrow MME)$	PKm	Asymmetric
$M5 (eNB2 \rightarrow UE)$	PKue	Asymmetric
$M6 (UE \rightarrow eNB2)$	FSKe2u	Symmetric
$M1' (UE \rightarrow eNB2)$	FSKe2u	Symmetric
$M2' (eNB2 \rightarrow UE)$	FSKe2u	Symmetric
$M3' (eNB2 \rightarrow MME)$	PKm	Asymmetric

Claim 10: The protocol achieves optimization of re-authenticate in X2-handover.

Proof: There is no need to perform a complete handover security process in the second handover between the same eNodeBs. The target eNodeBs can authenticate UE by

$e2ueAuth^*$; $e2ueAuth^* = \text{AuthFunc}(e2ueAuth \oplus RNue')$ and the UE can authenticate target eNodeBs by $Xe2ueAuth$; $Xe2ueAuth = \text{AuthFunc}(e2ueAuth \oplus RNe2')$. Thus, the next handover will be lighter than the first one, and this helps to improve computational cost and QoS level. Besides, in the second handover between the same eNodeBs (re-Authentication), all the AS and NAS keys including $KRRCint^*$, $KRRCenc^*$, $KRRCUPenc^*$, $KNASenc^*$, and $KNASint^*$ can be generated by their last versions ($KRRCint$, $KRRCenc$, $KRRCUPenc$, $KNASenc$, and $KNASint$). And also, an Updated Key (UK) is calculated by $RNe2'$, $RNue'$, and $e2ueAuth$; $UK = UKGFunc(RNe2', RNue', e2ueAuth)$. In other words, new keys in AS/NAS layers are based on their previous version, random number, and $e2ueAuth$. Accordingly, there is a significant decrease in computational cost second handover attempt. This can be helpful to have a better QoS level in the LTE system.

Claim 11: The REPS-AKA2 protocol achieves strong session key.

Proof: In the REPS-AKA2 protocol, source eNB employs a strong hash function to generate corresponding session key. Thus, the REPS-AKA2 can cover backward key separation goal properly. For each involved eNodeB, it is infeasible to extract the previous version of the related keys by using existing session keys (i.e., $KeNB$). Thus, if one eNodeB is corrupted by an attacker, without the last $KeNB$, the shared messages among the source eNodeBs and the onboard UEs will not be extracted.

Claim 12: The proposed REPS-AKA2 provides forward key separation.

Proof: To cover forward key separation in REPS-AKA2 method, the source eNodeB generates fresh session keys using the NKG value. The NKG is calculated by random

parameters (RNe_2 , RNe , and RNe_1); $NKG = NKGFunc(RNe_2, RNe, RNe_1)$. In other words, the source eNB makes a fresh key for the target eNB during the X2 handover. The attacker (i.e. rogue eNodeB) is not able to calculate the session keys. Therefore, a forward key separation goal is provided and a corrupted eNB is unable to compromise other eNBs.

The security of the REPS-AKA2 protocol is compared with some suggested methods as shown in Table 11. We can see that the scheme proposed in REPS-AKA2 protocol cover forward hop security and it can protect the system against de-synchronization attacks. Thus, compared with the standard EPS-AKA, and Ref. [18], the presented REPS-AKA2 protocol is more secure.

Table 11: Comparison of security goals in Handover methods

Security goals	EPS-AKA	Ref. [18]	REPS-AKA2
One-hop Forward Safety	No	Yes	Yes
Anti-de-synchronization attack	No	No	Yes

Moreover, one more security comparison of the handover protocols is indicated in Table 12. Comparison of security properties confirms that the suggested REPS-AKA2 approach achieve more security goals.

Table 12: Comparison of security properties in Handover schemes

Security properties	Ref. [86]	Ref. [49]	Ref. [50]	Ref. [6]	REPS-AKA2
SP1	No	No	Yes	No	Yes
SP2	Yes	Yes	Yes	Yes	Yes
SP3	No	Yes	No	No	Yes
SP4	No	No	Yes	No	Yes
SP5	No	Yes	Yes	No	Yes
SP6	No	Yes	Yes	No	Yes
SP7	No	Yes	Yes	Yes	Yes
SP8	No	Yes	Yes	Yes	Yes
Satisfied properties	1	6	7	3	8

SP1: Mutual authentication; SP2: Key agreement; SP3: Privacy preservation; SP4: Prevention from signaling congestion prevent; SP5: Overcome MiTM attack; SP6: Overcome impersonation attack; SP7: Overcome replay attack; SP8: Overcome DoS and insider attacks;

4.3 Performance analysis of the proposed model: REPS-AKA2

The improvement we suggested in the REPS-AKA2 inherits basic structure of key management in the X2 handover. The main important difference is that the proposed scheme uses a combination of Symmetric and Asymmetric key infrastructure and improves performance metrics especially in the second handover attempt (re-Authentication phase). The evaluation of the REPS-AKA2 method is presented in terms of the most important performance metrics.

4.3.1 Communication overhead

The Communication Overhead (CO) of the REPS-AKA2 is evaluated in this section. We suppose that the cost of handover message delivery between the eNodeB and the MME is β unit, between the UE and the eNodeB is μ unit, and between eNodeBs is δ unit. We assume that the μ is in this range: $0 < \mu < \beta$. The reason is that the MME usually locates far away from the eNodeBs, in addition, the μ and δ values are logically less than β . We compare the CO of REPS-AKA2 protocol with some other AKA protocols [4] [7] [13-18][20] as indicated in Table 13.

Table 13: Comparison of communication overheads

Schemes	S1 Interface (β)	Uu Interface (μ)	X2 Interface (δ)
<i>Ref. [13]</i>	8	13	10
<i>Ref. [14]</i>	2	7	0
<i>Ref. [15]</i>	0	4	4
<i>Ref. [16]</i>	0	5	2
<i>Ref. [17]</i>	0	4	4
<i>Ref. [18,20,4]</i>	4	3	2
<i>Original method [7]</i>	4	3	2
<i>REPS-AKA2 (Ph1)</i>	1	3	2
<i>REPS-AKA2 (Ph2)</i>	1	2	0

Our scheme outperforms all other schemes since REPS-AKA2 methods have the lowest *CO* as shown in Figure 25. The improvement percentages of the 1st and 2nd phases of the suggested REPS-AKA2 method over the other listed protocols are 81%, 33%, 25%, 14%, 25%, 33%, 33%, 33%, 33%, and 90%, 67%, 62%, 57%, 62%, 67%, 67%, 67%, 67%, respectively.

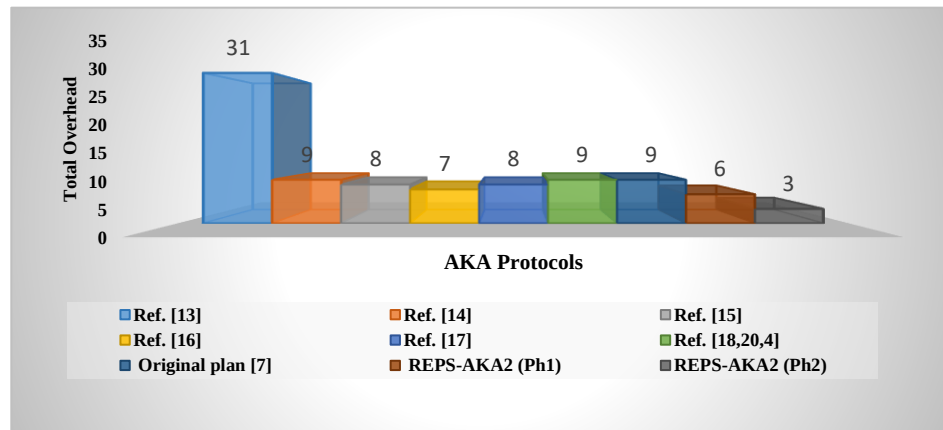


Figure 25: Comparison of communication overheads

4.3.2 Transmitted packets

We compare the number of Transmitted Packets (*TP*) between involved nodes in the proposed handover method with existing schemes as shown in Figure 26. The proposed REPS-AKA2 method requires fewer transmitted packets than Ref [56], [57],

[30], and [36]. In fact, the UEs have to send much more packets in other suggested methods which is not desirable because of bandwidth issues between the UEs and eNodeBs.

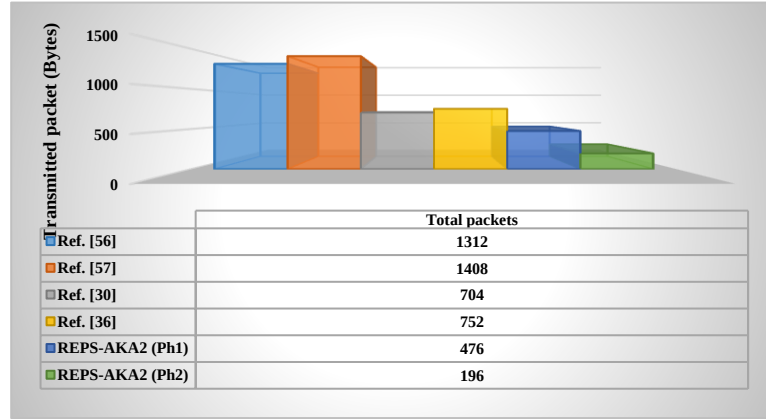


Figure 26: Comparison of TP in AKA methods

4.3.3 Computational cost

We evaluate Computation Cost (CC) of security operations in handover schemes. The computation cost of a bilinear pairing operator, a multiplication operator, an addition operator, and a hashing function are 4.14 ms, 0.86 ms, 0.58 ms, and 0.0065 ms respectively [36]. According, Table 14 shows the computation times of each LTE node. Table 14 concludes that the computation times in involved nodes in the proposed REPS-AKA2 are much fewer than other handover schemes.

Table 14: Computation times of involved nodes

Security Attribute	UE	eNB	MME
<i>Ref. [56]</i>	6.73	10.19	6.64
<i>Ref. [57]</i>	6.73	10.19	6.64
<i>Ref. [30]</i>	9.9	9.9	8.7
<i>Ref. [36]</i>	0.586	8.87	14.173
<i>REPS-AKA2 (Ph1)</i>	0.0715	0.052	0.0195
<i>REPS-AKA2 (Ph2)</i>	0.065	0.0455	0.013

Chapter 5

MODEL 3 (REPS-AKA3)

In this chapter, we present an enhanced AKA protocol named as REPS-AKA3 which can provide acceptable immunity against malicious attacks. In fact, the REPS-AKA3 is an enhanced version of the AKA protocol, which improves mutual authentication while needs some minor modifications to the LTE systems. The REPS-AKA3 method is evaluated and compared with the standard EPS-AKA scheme. This chapter is organized as follows: Section 5.1 demonstrates the proposed REPS-AKA3 protocol to solve security issues of LTE network. Next, we provide performance evaluation and security analysis of the REPS-AKA3 method in sections 5.2 and 5.3 respectively.

5.1 The proposed REPS-AKA3 protocol

To provide perfect security in the LTE systems, we suggest the REPS-AKA3 protocol that covers all security contexts. The proposed REPS-AKA3 can be divided into five phases/contexts as follows: Pre-authentication phase, EPS security Context, NAS Security Context (between MME and UE), AS Security Context (between UE and eNodeB), and Re-authentication phase.

5.1.1 Pre-authentication phase

The pre-authentication procedure is based on a group authentication mechanism. When a UE joins a given LTE network, it needs to attach to a legitimate eNodeB. Since there is more than one eNodeB in each LTE area, the UE may connect to a fake eNodeB (attacker). The UEs can connect to the legitimate eNodeBs using the pre-authentication phase of the REPS-AKA3 protocol. According to REPS-AKA3, each

eNodeB sends AKA Advertising Table (AAT) to UEs in its cell (i.e., eNodeB's zone), as shown in Figure 27. The AAT has public information about eNodeB and its neighbors, including eNodeB's ID, Public keys, and connected MME. Each eNodeB can communicate with some other eNodeBs via X2 interfaces and with the EPC via S1-C to make/update its AAT frequently.

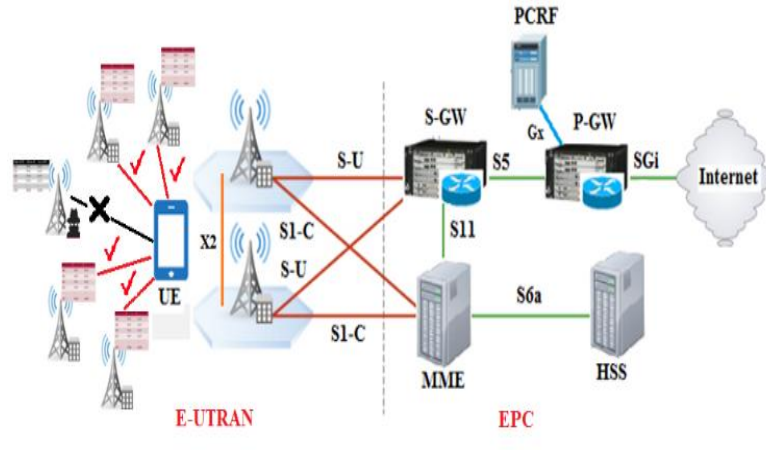


Figure 27: Fake and legitimate eNodeBs

An example of AAT is shown in Table 15. Neighbor eNodeB's ID (NeID) is a unique ID of each eNodeB located in the neighbor area. Connected MME ID (CMID) is a unique identity of the MME in the serving network. Moreover, the AAT includes the public key of eNodeB (PKenb) and connected MME (PKmme). In addition, the AATs have an identity (FeID) and public key (PKFenb) of fake eNodeBs.

Table 15: Example of AAT

Neighbor eNBs ID	Public key of eNB	Connected MME ID	Public key of MME
NeID 1	PKenb1	CMID1	PKmme1
NeID 2	PKenb2	CMID1	PKmme1
NeID 3	PKenb3	CMID2	PKmme2
NeID 4	PKenb4	CMID2	PKmme2
FeID 1	PKFenb1	FMID1	PKFmme1
FeID 2	PKFenb2	FMID2	PKFmme2

Finally, the UE merges the AATs and creates Final Decision Table (FDT), as shown in Table 16. The UE adds the signal level (SL) of eNodeBs to the FDT. The level of received signals can be Very Low (VL), Low (L), Medium (M), High (H), and Very High (VH). Count Number (CN) of an eNodeB is the total number of occurrences of that eNodeB in the received AATs. The number of previously Successful Attach (SA) is included in FDT. The UE lists eNodeBs based on SL, CN, and SA in FDT.

Table 16: Example of FDT

NeID	SL	CN	SA	PKenb	CMID	PKmme
NeID 1	VH	5	8	PKenb1	CMID1	PKmme1
NeID 2	VH	4	2	PKenb2	CMID1	PKmme1
NeID 3	H	4	7	PKenb3	CMID2	PKmme2
NeID 4	VH	2	6	PKenb4	CMID2	PKmme2
NeID 5	L	3	5	PKenb5	CMID3	PKmme3

An important point of REPS-AKA3 is that the UEs increase the count number of fake eNodeB when it is mentioned in AATs. And after successful AS context security, the UE sends its updated Suspicious Table (ST), including possible fake eNodeBs, to connected eNodeB. One example of ST is illustrated in Table 17.

Table 17: Example of ST

FeID	CN	PKFenb	FMID	PKFmme
FeID 1	5	PKFenb1	FMID1	PKFmme1
FeID 2	4	PKFenb2	FMID1	PKFmme1
FeID 3	4	PKFenb3	FMID2	PKFmme2
FeID 4	2	PKFenb4	FMID2	PKFmme2
FeID 5	3	PKFenb5	FMID3	PKFmme3

Where FeID, CN, PKFenb, FMID, and PKFmme are fake eNodeB identity, count number of fake eNodeB, the public key of fake eNodeB, fake MME identity, and public key of fake MME, respectively. The eNodeBs share a similar eNodeB

Information Table (EIT) table between neighbor eNodeBs periodically to update their database (i.e., fake and genuine eNodeBs). Thus, fake eNodeB frequently changes its information (i.e. NeID, PKenb, CMID, and PKmme). Thus, a fake eNodeB obtains new information in AATs. This decreases the count number of fake eNodeB in AAT. Consequently, the risk of incorrect attachment will be reduced considerably. One example of EIT is indicated in Table 18.

Table 18: Example of EIT

Neighbor eNBs ID	Public key of NB	Connected MME ID	Public key of MME
NeID 1	PKenb1	CMID1	PKmme1
NeID 2	PKenb2	CMID1	PKmme1
NeID 3	PKenb3	CMID2	PKmme2
NeID 4	PKenb4	CMID2	PKmme2
FeID 1	PKFenb1	FMID1	PKFmme1
FeID 2	PKFenb2	FMID2	PKFmme2

5.1.2 EPS security context

This section presents the EPS security context. First, we define the pre-shared keys and parameters as follows: **SK key**: The secret key (SK) is a main pre-shared key between HSS and UE at the time of manufacturing; **SKmh**: The SKmh is a secret key between the MME and the HSS. This key can be updated periodically; **SKem**: This is a pre-shared key for communication channels between eNodeB and MME; **MVP and HVP**: The MME Validation Parameter (MVP) and HSS Validation Parameter (HVP) are introduced as unique identifiers for MME and HSS. These identifiers can be changed after a pre-defined number of successful attach; **Public keys of MME and HSS**: MME and HSS share their public keys (PKmme, PKhss). In addition, the MME sends its public key to the UEs in AATs; **Key/parameter Generator Functions**: The REPS-AKA3 uses these cryptographic functions to generate various Keys/parameters for different layers (AS and NAS). Table 19 defines the role of these functions.

Table 19: Cryptographic functions

Functions	Roles	Functions	Roles
F1	To make MAC	HIGFunc	To make HI
F2	To make RES and XRES	NKGFunc	To make key NKG
F3	To make key CK	AUTHFunc	To make AUTN
F4	To make key IK	NKeNBG	To make key NKeNB
F5	To make key AK	LAFunc	To make LAUTHP
HFunc	To make HMSI	UKFunc	To generate key UK
UIGFunc	To make UI	NIFunc	To make HNID
KDF	To generate key KASME, KNASenc, and RRCUPenc,	UFunc	To generate URAND, UAUTH, and UKASME

A clear picture of the keys hierarchy and steps of REPS-AKA3 are shown in Figure 28 and 29 respectively.

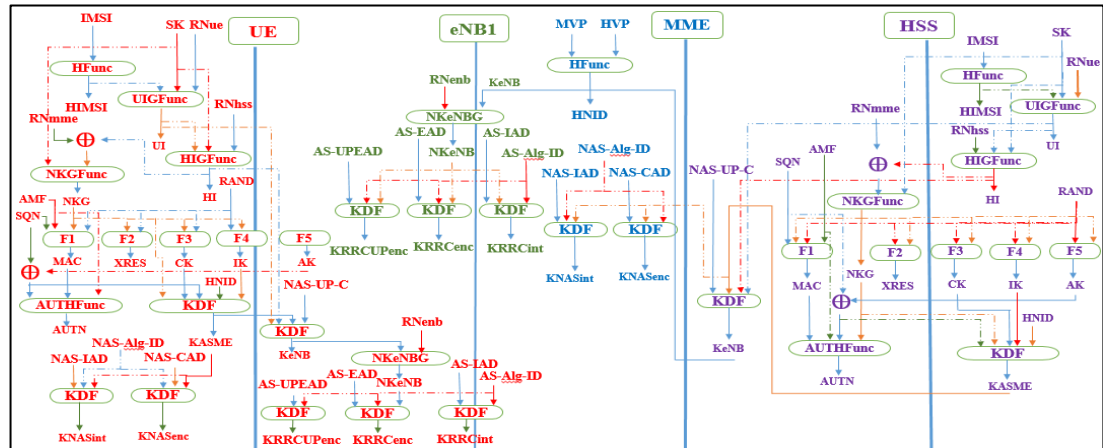


Figure 28: Keys generation model of REPS-AKA3 Protocol

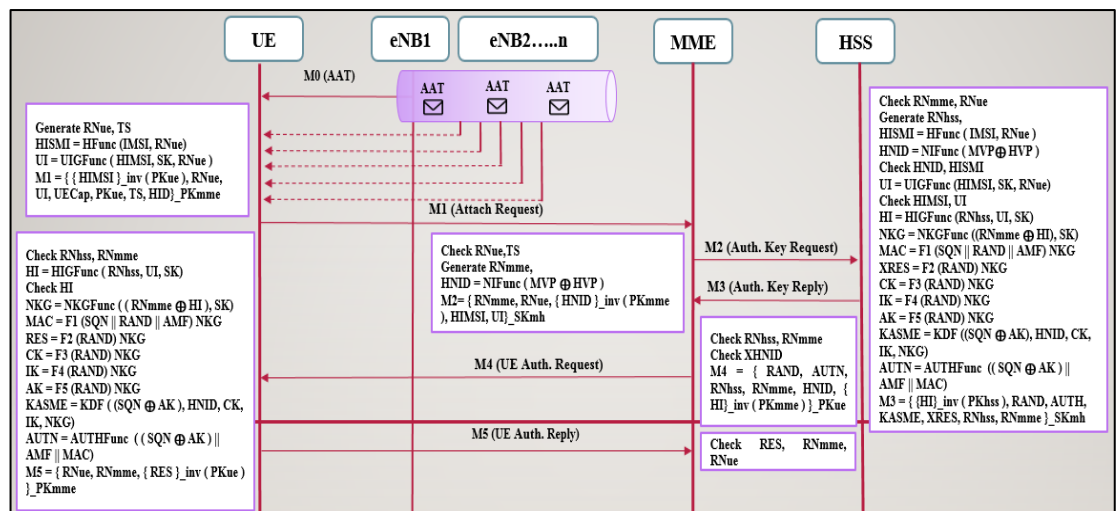


Figure 29: Steps of REPS-AKA3 (EPS security context)

The steps of REPS-AKA3 (EPS security context) are presented as follows (Figure 29):

1. UE → MME (M1: Attach Request): When an LTE user joins an LTE network for the first time, s/he needs to send UE's identify using an Attach Request. First, the UE generates a Random Number (RNue) and sets a timestamp (TS). This pair (RNue, TS) value will confirm that the current request message is fresh. In this manner, we can protect MME against replay attacks. Then, it executes the function HFunc on the IMSI to produce Hidden IMSI (HIMSI): $HIMSI = HFunc(IMSI, RNue)$. Thus, the IMSI will be encrypted with the random number (RNue) as a key that is never seen before. This can prevent replay attacks. In this way, we can be sure that IMSI is sent to MME securely. Next, the UE encrypts the HIMSI with its inverse public key (PKue) to prove that the message comes from a genuine UE. The UE also generates a User Identifier (UI) that works as a unique identifier between HSS and UE. The UE calculates a new UI using RNue, generated HIMSI, and its SK: $UI = UIGFunc(HIMSI, SK, RNue)$. Then, the UE sets UE Capability (UECap) and HSS identity (HID). The UECap shows the UE capability for encryption and decryption. The HID is the unique number that is HSS ID. Consequently, the UE transmits M1 (i.e., Attach Request) including {HIMSI}_inv (PKue), RNue, UI, UECap, PKue, TS, and HID. The message M1 is encrypted by PKmme.

2. MME → HSS (M2: Auth. Key Request): The MME decrypts M1 with its private key and checks RNue and TS for freshness. Next, the MME in the serving network uses the HID to find the HSS's secret key stored in its database. Then, the MME makes Home Network Identity (HNID) using MVP and HVP: $HNID = NIFunc(MVP \oplus HVP)$. The HNID can help the system to have a unique value between a given MME and HSS. First, the MME encrypts the HNID using its inverse public key (PKmme) to

prove that the message comes from MME. After that, the MME generates RNmme and makes M2 (i.e., Auth. Key Request) including RNmme, RNue, {HNID}_inv (PKmme), HMSI, and UI. Finally, the MME encrypts M2 by a secret key of MME and HSS (SKmh) and sends it to HSS.

3. HSS → MME (M3:Auth. Key Reply): In the home network, the HSS decrypts the received message by its secret key (SKmh) and extracts the values of RNmme, RNue, HNID, HMSI, and UI. To authenticate the MME, the HSS calculates HNID: $HNID = NIFunc(MVP \oplus HVP)$ and compares this value with the received one (HNID). When MME is successfully authenticated, the HSS applies HMSI and RNue to the HFunc function and generates HMSI: $HMSI = HFunc(HMSI, RNue)$. The HSS uses the generated HMSI, received RNue and SK of UE to calculate UI value: $UI = UIGFunc(HMSI, SK, RNue)$ and compares it with the received UI to verify UE. Thus, the HSS confirms the UE and the MME by checking UI and HNID, respectively. If the UE and MME are genuine nodes, then HSS makes a random number (RNhss) and produces HSS Identifier (HI) using RNhss, UI, SK: $HI = HIGFunc(RNhss, UI, SK)$. The HI parameter is a unique value between UE and HSS. The UE uses the HI value to authenticate HSS. The HSS forwards HI, encrypted by its inverse public key (PKhss), to the message authentication procedure. Moreover, the HSS calculates New Key Generator (NKG) based on RNmme, HI, and SK: $NKG = NKGFunc((RNmme \oplus HI), SK)$. Then, the HSS generates random value (RAND). It uses the RAND, SQN, AMF, HNID, and NKG to generate MAC, XRES, CK, IK, AK, KASME, and AUTN: $MAC = F1(SQN \parallel RAND \parallel AMF) NKG$, $XRES = F2(RAND) NKG$, Cipher Key (CK) = $F3(RAND) NKG$, Integrity Key (IK) = $F4(RAND) NKG$, Anonymity Key (AK) = $F5(RAND) NKG$, $KASME = KDF((SQN \oplus AK), HNID, CK, IK, NKG)$,

$AUTN = AUTHFunc ((SQN \oplus AK) \parallel AMF \parallel MAC)$. The HSS encrypts M3 (i.e. Auth. Key Reply) message by secret key (SKmh) and forwards M3 to the MME. The Auth. Key Reply message includes $\{HI\}_{inv}$ (PKhss), RAND, AUTN, KASME, XRES, RNhss, and RNmme.

4. MME → UE (M4: UE Auth. Request): The MME in serving network employs its symmetric key (SKmh) to decrypt the received message and authenticates the HSS by checking the received RNmme. The MME also authenticates the received message by $\{HI\}_{inv}$ (PKhss). After successful authentication of the HSS, serving MME stores received data and sends M4 (i.e., UE Auth. Request), which is encrypted by UE's public key (PKue). The M4 includes RAND, AUTN, RNhss, RNmme, HNID, and $\{HI\}_{inv}$ (PKmme).

5. UE → MME (M5: UE Auth. Reply): The UE decrypts the message M4 by using its private key and checks the RNhss and RNmme for freshness. Then, the UE generates HI by received RNhss, generated UI, and its secret key (SK): $HI = HIFunc (RNhss, UI, SK)$ and compares generated HI with received HI to authenticate HSS. After successfully authenticating HSS, the UE calculates the value of the NKG key using RNmme, HI, and SK: $NKG = NKGFunc ((RNmme \oplus HI), SK)$. The UE uses RAND, SQN, AMF, HNID, and NKG to calculate MAC, RES, CK, IK, AK, KASME, and AUTN: $MAC = F1 (SQN \parallel RAND \parallel AMF) NKG$, $RES = F2 (RAND) NKG$, $CK = F3 (RAND) NKG$, $IK = F4 (RAND) NKG$, $AK = F5 (RAND) NKG$, $KASME = KDF ((SQN \oplus AK), HNID, CK, IK, NKG)$, $AUTN = AUTHFunc ((SQN \oplus AK) \parallel AMF \parallel MAC)$. The UE utilizes the AUTN to authenticate the network side. Moreover, it uses KASME as a local master key to generate security keys in AS and NAS layers.

The UE forwards M5 (i.e., UE Auth. Reply) including RNue, RNmme, and {RES}_inv (PKue) to MME. The message (M5) is encrypted by PKmme.

6. MME: The MME extracts RNue, RNmme, and RES from message M5 by its private key. Then, it checks RNue, and RNmme for freshness and compares the received RES with saved XRES to confirm the UE. In addition, the MME can authenticate received messages by {RES}_inv (PKue). If these matches are successful, the MME can start NAS security context. Otherwise, the MME forwards a failure message to the UE.

5.1.3 NAS Security context

The REPS-AKA3 protocol steps (NAS Security Context) are presented in Figure 30.

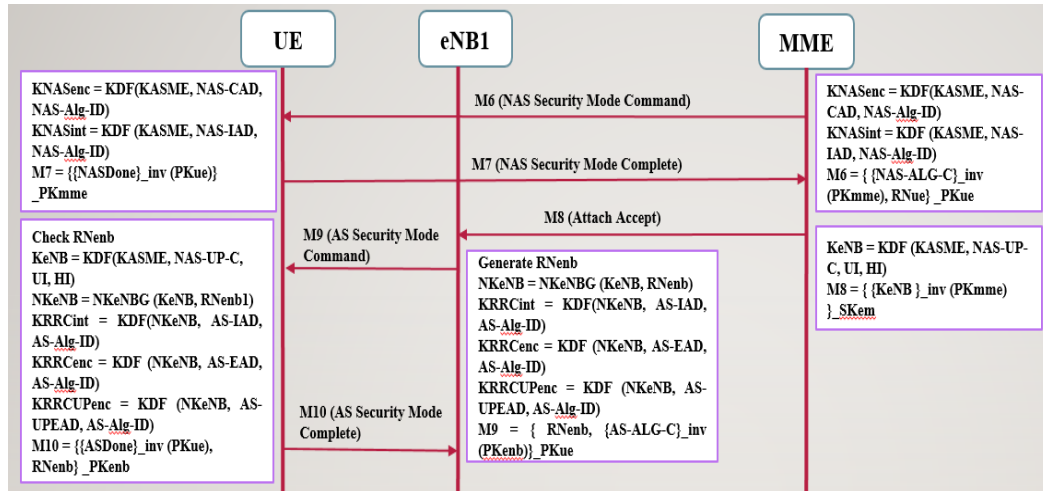


Figure 30: Steps of REPS-AKA3 (NAS/AS Security Context)

1. MME→UE (M6: NAS Security Mode Command): The MME selects security algorithms of the NAS layer according to the UECap parameter to generate NAS security keys. The MME uses KDF function, KASME, NAS Ciphering Algorithm Distinguisher (NAS-CAD), NAS Integrity Algorithm Distinguisher (NAS-IAD), and NAS Algorithm ID (NAS-Alg-ID) to calculate key NAS Encryption (KNASenc) and

key NAS Integrity (KNASint): $KNAS_{enc} = KDF(KASME, NAS-CAD, NAS-Alg-ID)$, $KNAS_{int} = KDF(KASME, NAS-IAD, NAS-Alg-ID)$. Then, the MME sets NAS Algorithm Category (NAS-ALG-C). The MME encrypts the NAS-ALG-C value by its inverse public key (PK_{mme}) to prove that the message comes from genuine MME surely. The UE selects NAS security algorithms based on transmitted NAS-ALG-C. The MME forwards message M6 to the UE encrypted by PK_{ue}. The M6 includes {NAS-ALG-C}_inv (PK_{mme}) and RN_{ue}.

2. UE → MME (M7: NAS Security Mode Complete): The UE decrypts M6 and extracts {NAS-ALG-C}_inv (PK_{mme}) and RN_{ue} to be sure about message freshness and authentication. The UE also uses NAS-ALG-C to select NAS security algorithms and calculates security keys in the NAS layer. Then, the UE employs KASME, NAS-CAD, NAS-IAD, and NAS-Alg-ID to calculate $KNAS_{enc}$ and $KNAS_{int}$ for NAS Security Context: $KNAS_{enc} = KDF(KASME, NAS-CAD, NAS-Alg-ID)$, $KNAS_{int} = KDF(KASME, NAS-IAD, NAS-Alg-ID)$. The UE sends M7 (i.e. NAS Security Mode Complete) including {NASDone}_inv (PK_{ue}) encrypted by PK_{mme}. This confirmation value (NASDone) is encrypted by UE's inverse public key (PK_{ue}) to prove that the message come from genuine UE surely. By sending message M7, the UE completes procedure of NAS security context.

5.1.4 AS Security context

The AS Security Context is created when the UE is in a “connected” state as shown in Figure 30. The keys of AS layer are derived by using KASME and main session key (KeNB). The REPS-AKA3 protocol steps in the AS Security Context are as follows:

1. MME → eNodeB (M8: Attach Accept): For starting the AS security Context, the MME generates a session key (i.e., KeNB) using KASME, UI, HI, and NAS Uplink

Count (NAS-UP-C): $KeNB = KDF(KASME, NAS-UP-C, UI, HI)$. Then, the MME encrypts the calculated KeNB using its inverse public key (PKmme) to prove that the message comes from genuine MME. Finally, the MME forwards calculated KeNB to eNodeB by message M8 (i.e., Attach Accept). The M8 is encrypted by the symmetric key between MME and eNodeB (SKem).

2. eNodeB → UE (M9: AS Security Mode Command): The eNodeB decrypts M8 and extracts the KeNB. Then, it generates a random value (R_{Nenb}) and computes New KeNB (NKeNB) using KeNB and NKeNBG function: $NKeNB = NKeNBG(KeNB, R_{Nenb})$. Then, the eNodeB selects AS Algorithm ID (AS-Alg-ID) to calculate dependent keys for AS Security Context. It uses calculated NKeNB, AS-Alg-ID, AS Integrity Algorithm Distinguisher (AS-IAD), AS Encryption Algorithm Distinguisher (AS-EAD), and AS User Plane Encryption Algorithm Distinguisher (AS-UPEAD) to make Key RRC Integrity (KRR_{Cint}), Key RRC Cipher (KRR_{Cenc}), and Key RRC User Plane cipher (KRR_{CUPenc}): $KRR_{Cint} = KDF(NKeNB, AS-IAD, AS-Alg-ID)$, $KRR_{Cenc} = KDF(NKeNB, AS-EAD, AS-Alg-ID)$, and $KRR_{CUPenc} = KDF(NKeNB, AS-UPEAD, AS-Alg-ID)$. Then, the eNodeB sets AS Algorithm Category (AS-ALG-C) and forwards it to UE. The UE selects security functions in AS layer according to the transmitted AS-ALG-C. The eNodeB forwards M9 including R_{Nenb}, {AS-ALG-C}_inv (PK_{enb}) to UE. The message M9 is encrypted by public key of UE (PK_{ue}).

3. UE → eNodeB (M10: AS Security Mode Complete): When UE receives M9, it extracts R_{Nenb}, AS-ALG-C by its private key. Then, the UE generates KeNB: $KeNB = KDF(KASME, NAS-UP-C, UI, HI)$. The UE uses KeNB and received R_{Nenb} to calculate NKeNB: $NKeNB = NKeNBG(KeNB, R_{Nenb1})$. After that, the UE employs NKeNB, AS-IAD, AS-EAD, AS-UPEAD, and AS-Alg-ID to compute KRR_{Cint},

KRRCenc, and KRRCUPenc: $KRRCint = KDF(NKeNB, AS-IAD, AS-Alg-ID)$, $KRRCenc = KDF(NKeNB, AS-EAD, AS-Alg-ID)$, and $KRRCUPenc = KDF(NKeNB, AS-UPEAD, AS-Alg-ID)$. The UE sets and encrypts ASDone by its inverse public key (PKue) to prove that this message comes from genuine UE. The UE forwards M10 (i.e., AS Security Mode Complete) to eNodeB encrypted by PKenb. Message M10 includes {ASDone}_inv (PKue) and RNenb.

5.1.5 Re-authentication procedure

If one UE is located in the serving network for a long time, it may send its n^{th} authentication request to the MME. In this case, the LTE nodes can use previously shared information to perform AKA protocol. Thus, when the subscriber wants a re-attach to the network, a re-authentication phase of REPS-AKA3 starts to execute as shown in Figure 31. The steps of re-authentication phase are as follows:

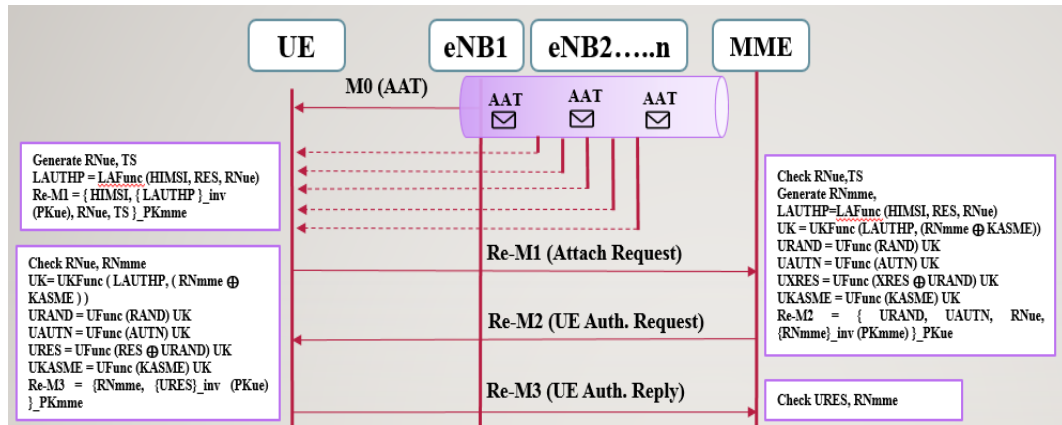


Figure 31: Steps of REPS-AKA3 (Re-authentication)

1. UE → MME (Re-M1: Attach Request): The UE generates a new Random Number (RNue) and sets a new Timestamp (TS). This pair (RNue, TS) is for freshness. Then, the UE employs LAFunc, and related parameters (i.e., HIMSI, RES and RNue) to generate the Last Authentication Parameter (LAUTHP): $LAUTHP = LAFunc(HIMSI, RES, RNue)$. The UE forwards LAUTHP to the MME to prove that the UE is

registered to MME previously, and this is at least 2nd authentication. The UE forwards Re-M1 (i.e., Attach Request) encrypted by the public key of MME (PK_{mme}). The Re-M1 includes {HIMSI, {LAUTHP}_{inv} (PK_{ue}), RN_{ue}, and TS.

2. MME → UE (Re-M2: UE Auth. Request): The serving MME decrypts the received Re-M1 (Attach Request) by its private key. The MME uses HIMSI, RES, and RN_{ue} to calculate LAUTHP: $LAUTHP = LA_{Func}(HIMSI, RES, RN_{ue})$. It authenticates the UE by verifying the received LAUTHP. Then, the MME generates a new random number (RN_{mme}) and Updated Key (UK): $UK = UK_{Func}(LAUTHP, (RN_{mme} \oplus KASME))$. The MME employs UK to generate Updated RAND (URAND), Updated AUTN (UAUTN), Updated XRES (UXRES), and Updated KASME (UKASME): $URAND = U_{Func}(RAND) UK$, $UAUTN = U_{Func}(AUTN) UK$, $UXRES = U_{Func}(XRES \oplus URAND) UK$, and $UKASME = U_{Func}(KASME) UK$. The MME forwards Re-M2 (UE Auth. Request) to UE encrypted by PK_{ue}. The Re-M2 includes URAND, UAUTN, RN_{ue}, and {RN_{mme}}_{inv} (PK_{mme}).

3. UE → MME (Re-M3: UE Auth. Reply): The UE decrypts Re-M2 (i.e., UE Auth. Request) and checks RN_{mme} for freshness. Then, the UE generates the UK using received RN_{mme}, generated LAUTHP, and KASME: $UK = UK_{Func}(LAUTHP, (RN_{mme} \oplus KASME))$. The UE utilizes the UK to calculate URAND, UAUTN, URES, and UKASME: $URAND = U_{Func}(RAND) UK$, $UAUTN = U_{Func}(AUTN) UK$, $URES = U_{Func}(RES \oplus URAND) UK$, and $UKASME = U_{Func}(KASME) UK$. The remaining parts of the key agreement procedure are the same as the first attach. Then, the UE forwards Re-M3 (UE Auth. Reply) including RN_{mme}, {URES}_{inv} (PK_{ue}) to MME. Finally, the Re-M3 is encrypted by PK_{mme}.

4. MME: The MME extracts RNmme and URES from message Re-M3 by its private key. It compares the received URES with generated UXRES to verify the UE. The MME can authenticate received message by {URES}_inv (PKue). If these matches are successful, the MME can start NAS security context. The remaining parts of the key agreement procedure are the same as NAS and AS contexts in the first attach.

5.2 Security analysis of REPS-AKA3 protocol

In this part of the chapter, we provide a security analysis of suggested method by using BAN Logic, Scyther, and AVISPA tools to show that the suggested REPS-AKA3 method can achieve security goals properly.

5.2.1 Formal analysis using AVISPA

We have tested the proposed REPS-AKA3 protocol by using the AVISPA tool to check its security properties. The presented method provides secrecy goal to ensure that transmitted information is secret between defined entities. To verify the REPS-AKA3 protocol, we define secrecy and strong authentication goals as shown in Table 20. (AVISPA codes are presented in Appendix 1).

Table 20: Security goals of REPS-AKA3

Goal Name in AVISPA	Role
<i>authentication_on mme_ue_msg_auth_HIMSI</i>	MME authenticated UE's message by HIMSI
<i>authentication_on ue_mme_msg_auth_HI</i>	UE authenticated MME's message by HI
<i>authentication_on mme_ue_msg_auth_RES</i>	MME authenticated UE's message by RES
<i>authentication_on mme_ue_auth_RNmme</i>	MME authenticated UE by RNmme
<i>authentication_on ue_mme_msg_auth_NASALGC</i>	UE authenticated MME's message by NASALGC
<i>authentication_on ue_mme_auth_RNue</i>	UE authenticated MME by RNue
<i>authentication_on mme_ue_msg_auth_DONENAS</i>	MME authenticated UE's message by DONENAS
<i>authentication_on ue_enb_msg_auth_ASALGC</i>	UE authenticated eNB's message by ASALGC
<i>authentication_on enb_ue_msg_auth_DONEAS</i>	eNB authenticated UE's message by DONEAS
<i>authentication_on enb_ue_auth_RNenb</i>	eNB authenticated UE by RNenb
<i>authentication_on hss_mme_msg_auth_HNID</i>	HSS authenticated MME's message by HNID
<i>authentication_on mme_hss_auth_RNmme</i>	MME authenticated HSS by RNmme
<i>authentication_on mme_hss_msg_auth_HI</i>	MME authenticated HSS's message by HI
<i>authentication_on ue_mme_msg_auth_HI</i>	UE authenticated MME's message by HI
<i>authentication_on enb_mme_msg_auth_KeNB</i>	eNB authenticated MME's message by KeNB

secrecy_of ue_mme_HIMSI/ HID/ TS/ UI/ RNue/ RNmme/ RES/ DONENAS	HIMSI/ HID/TS/ UI/ RNue/ RNmme/ RES/ DONENAS are transmitted from UE to MME securely
secrecy_of ue_enb_RNenb/ DONEAS	RNenb/DONEAS are transmitted from UE to eNB securely
secrecy_of mme_hss_HNID/ HIMSI/ UI/ RNmme/ RNue	HNID/HIMSI/UI/RNmme/RNue are transmitted from MME to HSS securely
secrecy_of mme_ue_RAND/ AUTH/ RNhss/ RNmme/ HNID/ HI/ RNue/ NASALGC	RAND/AUTH/RNhss/RNmme/NASALGC are transmitted from MME to UE securely
secrecy_of mme_enb_KeNB	KeNB is transmitted from MME to eNB securely
secrecy_of hss_mme_HI/ RAND/ AUTH/ KASME/ XRES/ RNhss/ RNmme	HI/RAND/KASME/XRES/RNhss/RNmme are transmitted from HSS to MME securely
secrecy_of enb_ue_RNenb/ ASALGC	RNenb/ ASALGC are transmitted from eNB to UE securely

The HPSL code of the proposed protocol is divided into seven parts. The first four parts define the roles of the involved entities, including UE, MME, HSS, and eNodeB. The last three parts are the sessions, environment and goals of the protocol. The OFMC and CL-AtSe backends are used to validate the REPS-AKA3 protocol. According to the test results in Figure 32 and Figure 33 (Details are in Appendix1), “SAFE” keyword indicates that the modeled protocol is safe based on the specified goals. Thus, we can conclude that all security goals mentioned in Table 20 have been satisfied.

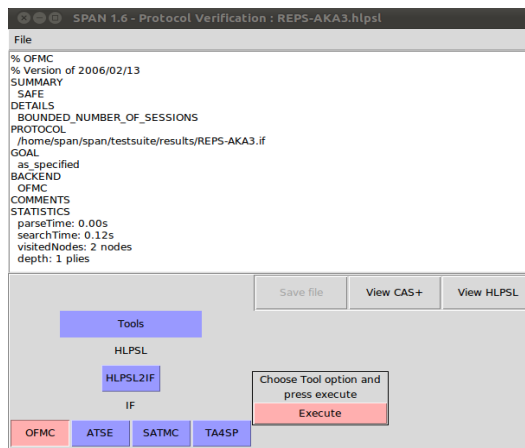


Figure 32: Evaluation result in OFMC

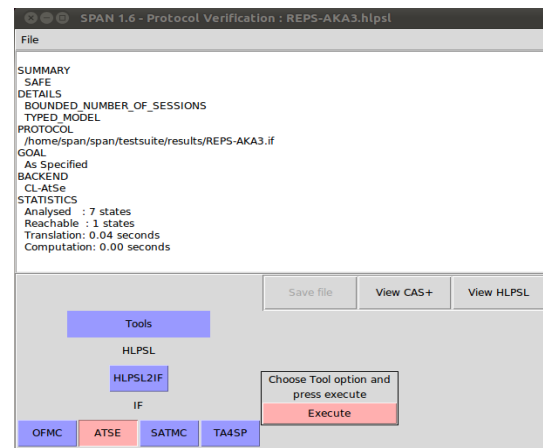


Figure 33: Evaluation result in CL-Atse

5.2.2 Formal analysis using Scyther

In this section, the proposed REPS-AKA3 method is evaluated by means of Scyther tool. The description of the presented REPS-AKA3 model is implemented based on the defined nodes (i.e., UE, eNodeB, MME, and HSS). In the modeled protocol, five claims (i.e., Nisynch, Secrecy, Weakagree, Niagree, and Alive) are checked for mentioned nodes. (Scyther codes are presented in Appendix 2).

Based on UE's viewpoint in REPS-AKA3, the security claims are presented as follows: *claim_ue1(UE, Secret, HMSI); claim_ue2 (UE, Secret, UI); claim_ue3 (UE, Secret, UECap); claim_ue4 (UE, Secret, TS); claim_ue5 (UE, Secret, HID); claim_ue6 (UE, Secret, RNue); claim_ue7 (UE, Secret, RNmme); claim_ue8 (UE, Secret, RES); claim_ue9 (UE, Secret, NASDONE); claim_ue10 (UE, Secret, ASDONE); claim_ue11(UE, Secret, LAUTHP); claim_ue12(UE, Secret, URES); claim_uea (UE, Alive); claim_uew(UE, Weakagree); claim_uenia(UE, Niagree); claim_uenis(UE, Nisynch);*

Based on eNodeB's viewpoint in REPS-AKA3, the security claims are presented as follows: *claim_enb1 (ENB, Secret, RNenb); claim_enb2 (ENB, Secret, ASALGC); claim_enba (ENB, Alive); claim_enbw (ENB, Weakagree); claim_enbnia (ENB, Niagree); claim_enbnis (ENB, Nisynch);*

Based on MME's viewpoint in REPS-AKA3, the security claims are presented as follows: *claim_mme1 (MME, Secret, RNmme); claim_mme2 (MME, Secret, RNue); claim_mme3 (MME, Secret, HNID); claim_mme4 (MME, Secret, HMSI); claim_mme5 (MME, Secret, UI); claim_mme6 (MME, Secret, RAND); claim_mme7 (MME, Secret, AUTN); claim_mme8 (MME, Secret, RNhss); claim_mme9 (MME, Secret, NASALGC); claim_mme10 (MME, Secret, KENB); claim_mme11 (MME, Secret, URAND); claim_mmea (MME, Alive); claim_mmew (MME, Weakagree); claim_mmenia (MME, Niagree); claim_mmenis (MME, Nisynch);*

Based on HSS's viewpoint in REPS-AKA3, the security claims are presented as follows: *claim_hss1 (HSS, Secret, HI); claim_hss2 (HSS, Secret, RAND); claim_hss3 (HSS, Secret,*

AUTN); *claim_hss4* (*HSS*, *Secret*, *KASME*); *claim_hss5* (*HSS*, *Secret*, *XRES*); *claim_hss6* (*HSS*, *Secret*, *RNhss*); *claim_hss7* (*HSS*, *Secret*, *RNmme*); *claim_hssa* (*HSS*, *Alive*); *claim_hssw* (*HSS*, *Weakagree*); *claim_hssnia* (*HSS*, *Niagree*); *claim_hssnis* (*HSS*, *Nisynch*);

The evaluation results of REPS-AKA3 protocol are presented in Figures 34-37. As it can be seen in these Figures, the proposed protocols satisfy all defined security climes in involved nodes (UE, MME, HSS and eNodeB).

Claim				Status		Comments
REPSAKA3	HSS	REPSAKA3,hss1	Secret HIGFunc(RNhss,UIGFunc(HFunc(IMS1,RNue),SK,R...	Ok	Verified	No attacks.
		REPSAKA3,hss2	Secret RAND	Ok	Verified	No attacks.
		REPSAKA3,hss3	Secret AUTHFunc(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,...	Ok	Verified	No attacks.
		REPSAKA3,hss4	Secret KDF(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,HIGFu...	Ok	Verified	No attacks.
		REPSAKA3,hss5	Secret F2(RAND,NKGFunc(XOR(RNmme,HIGFunc(RNhss,UIG...	Ok	Verified	No attacks.
		REPSAKA3,hss6	Secret RNhss	Ok	Verified	No attacks.
		REPSAKA3,hss7	Secret RNmme	Ok	Verified	No attacks.
		REPSAKA3,hssa	Alive	Ok	Verified	No attacks.
		REPSAKA3,hssw	Weakagree	Ok	Verified	No attacks.
		REPSAKA3,hssnia	Niagree	Ok	Verified	No attacks.
		REPSAKA3,hssnis	Nisynch	Ok	Verified	No attacks.

Figure 34: Scyther evaluation for defined claims in the HSS (REPS-AKA3)

Claim				Status		Comments
REPSAKA3	UE	REPSAKA3,ue1	Secret HFunc(IMS1,RNue)	Ok	Verified	No attacks.
		REPSAKA3,ue2	Secret UIGFunc(HFunc(IMS1,RNue),SK,RNue)	Ok	Verified	No attacks.
		REPSAKA3,ue3	Secret UECap	Ok	Verified	No attacks.
		REPSAKA3,ue4	Secret TS	Ok	Verified	No attacks.
		REPSAKA3,ue5	Secret HIID	Ok	Verified	No attacks.
		REPSAKA3,ue6	Secret RNue	Ok	Verified	No attacks.
		REPSAKA3,ue7	Secret RNmme	Ok	Verified	No attacks.
		REPSAKA3,ue8	Secret F2(RAND,NKGFunc(XOR(RNmme,HIGFunc(RNhss,UIG...	Ok	Verified	No attacks.
		REPSAKA3,ue9	Secret NASDONE	Ok	Verified	No attacks.
		REPSAKA3,ue10	Secret ASDONE	Ok	Verified	No attacks.
		REPSAKA3,ue11	Secret LAFunc(HFunc(IMS1,RNue),F2(RAND,NKGFunc(XOR...	Ok	Verified	No attacks.
		REPSAKA3,ue12	Secret UFunc(XOR(F2(RAND,NKGFunc(XOR(RNmme,HIGFunc...	Ok	Verified	No attacks.
		REPSAKA3,uea	Alive	Ok	Verified	No attacks.
		REPSAKA3,uew	Weakagree	Ok	Verified	No attacks.
		REPSAKA3,uenia	Niagree	Ok	Verified	No attacks.
		REPSAKA3,uenis	Nisynch	Ok	Verified	No attacks.

Figure 35: Scyther evaluation for defined claims in the UE (REPS-AKA3)

Claim				Status		Comments
REPSAKA3	MME	REPSAKA3,mme1	Secret RNmme	Ok	Verified	No attacks.
		REPSAKA3,mme2	Secret RNue	Ok	Verified	No attacks.
		REPSAKA3,mme3	Secret NIFunc(XOR(MVP,HVP))	Ok	Verified	No attacks.
		REPSAKA3,mme4	Secret HFunc(IMSI,RNue)	Ok	Verified	No attacks.
		REPSAKA3,mme5	Secret UIGFunc(HFunc(IMSI,RNue),SK,RNue)	Ok	Verified	No attacks.
		REPSAKA3,mme6	Secret RAND	Ok	Verified	No attacks.
		REPSAKA3,mme7	Secret AUTHFunc(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,...	Ok	Verified	No attacks.
		REPSAKA3,mme8	Secret RNhss	Ok	Verified	No attacks.
		REPSAKA3,mme9	Secret NASALGC	Ok	Verified	No attacks.
		REPSAKA3,mme10	Secret KDF(KDF(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,H...	Ok	Verified	No attacks.
		REPSAKA3,mme11	Secret UFunc(RAND,UKFunc(LAFunc(HFunc(IMSI,RNue),F...	Ok	Verified	No attacks.
		REPSAKA3,mme12	Secret UFunc(AUTHFunc(XOR(SQN,F5(RAND,NKGFunc(XOR(...	Ok	Verified	No attacks.
		REPSAKA3,mmea	Alive	Ok	Verified	No attacks.
		REPSAKA3,mmeu	Weakagree	Ok	Verified	No attacks.
		REPSAKA3,mmenia	Niagree	Ok	Verified	No attacks.
		REPSAKA3,mmenis	Nisynch	Ok	Verified	No attacks.

Figure 36: Scyther evaluation for defined claims in the MME (REPS-AKA3)

Claim				Status		Comments
REPSAKA3	ENB	REPSAKA3,enb1	Secret RNenb	Ok	Verified	No attacks.
		REPSAKA3,enb2	Secret ASALGC	Ok	Verified	No attacks.
		REPSAKA3,enba	Alive	Ok	Verified	No attacks.
		REPSAKA3,enbw	Weakagree	Ok	Verified	No attacks.
		REPSAKA3,enbnia	Niagree	Ok	Verified	No attacks.
		REPSAKA3,enbnis	Nisynch	Ok	Verified	No attacks.

Figure 37: Scyther evaluation for defined claims in the eNB (REPS-AKA3)

5.2.3 Security analysis by BAN logic

We prove the security of the presented REPS-AKA3 by using the BAN Logic. This security logic is an approach to perform a security verification of AKA protocols. The BAN logic properly verifies the transmitted messages and checks confidentiality among involved nodes in modeled AKA method.

The formal messages of the proposed protocol (REPS-AKA3) are as follows: (**M1: Attach Request**): $UE \rightarrow MME: \{\{HIMSI\}_{inv(PKue)}, RNue, UI, UECap, PKue, TS, HID\}_{PKmme}$, (**M2: Auth. Key Request**): $MME \rightarrow HSS: \{RNmme, RNue, \{HNID\}_{inv(PKmme)}, HIMSI, UI\}_{SKmh}$,

(M3:Auth. Key Reply): HSS $\rightarrow$ MME: $\{\{HI\}_{inv}(PK_{hss}), RAND, AUTN, KASME, XRES, RN_{hss}, RN_{mme}\}_{SK_{mh}}$, (M4: UE Auth. Request): MME $\rightarrow$ UE: $\{RAND, AUTN, RN_{hss}, RN_{mme}, HNID, \{HI\}_{inv}(PK_{mme})\}_{PK_{ue}}$, (M5: UE Auth. Reply): UE $\rightarrow$ MME: $\{RN_{ue}, RN_{mme}, \{RES\}_{inv}(PK_{ue})\}_{PK_{mme}}$, (M6: NAS Security Mode Command): MME $\rightarrow$ UE: $\{\{NAS-ALG-C\}_{inv}(PK_{mme}), RN_{ue}\}_{PK_{ue}}$, (M7: NAS Security Mode Complete): UE $\rightarrow$ MME: $\{\{NASDONE\}_{inv}(PK_{ue})\}_{PK_{mme}}$, (M8: Attach Accept): MME $\rightarrow$ eNodeB: $\{\{KENB\}_{inv}(PK_{mme})\}_{SK_{em}}$, (M9: AS Security Mode Command): eNodeB $\rightarrow$ UE: $\{RN_{enb}, \{AS-ALG-C\}_{inv}(PK_{enb})\}_{PK_{ue}}$, (M10: AS Security Mode Complete): UE $\rightarrow$ eNodeB: $\{\{ASDONE\}_{inv}(PK_{ue})\}_{PK_{enb}}$.

We need to transform all messages into idealized logical format as follows: (M1: Attach Request): UE $\rightarrow$ MME: $MME \triangleleft \{UE \mid \Rightarrow \{\{HIMSI\}PK_{ue}^{-1}, \#(RN_{ue}), UI, \#(TS), UECap, HID\}\}_{PK_{mme}}$, (M2: Auth. Key Request): MME $\rightarrow$ HSS: $HSS \triangleleft \{MME \mid \Rightarrow \{\#(RN_{mme}), \{HNID\}PK_{mme}^{-1}\}, UE \mid \Rightarrow \{\#(RN_{ue}), HIMSI, UI\}\}_{SK_{mh}}$, (M3:Auth. Key Reply): HSS $\rightarrow$ MME: $MME \triangleleft \{HSS \mid \Rightarrow \{\{HI\}PK_{hss}^{-1}, \#(RAND), AUTN, KASME, XRES, \#(RN_{hss})\}, MME \mid \Rightarrow \#(RN_{mme})\}_{SK_{mh}}$, (M4: UE Auth. Request): MME $\rightarrow$ UE: $UE \triangleleft \{HSS \mid \Rightarrow \{\#(RAND), AUTN, \#(RN_{hss}), \{HI\}PK_{mme}^{-1}\}, MME \mid \Rightarrow \{HNID, \#(RN_{mme})\}\}_{PK_{ue}}$, (M5: UE Auth. Reply): UE $\rightarrow$ MME: $MME \triangleleft \{UE \mid \Rightarrow \{\#(RN_{ue}), \{RES\}PK_{ue}^{-1}\}, MME \mid \Rightarrow \#(RN_{mme})\}_{PK_{mme}}$, (M6: NAS Security Mode Command): MME $\rightarrow$ UE: $UE \triangleleft \{MME \mid \Rightarrow \{NAS-ALG-C\}PK_{mme}^{-1}, UE \mid \Rightarrow \#(RN_{ue})\}_{PK_{ue}}$, (M7: NAS Security Mode Complete): UE $\rightarrow$ MME: $MME \triangleleft \{\{UE \mid \Rightarrow NASDONE\}PK_{ue}^{-1}\}_{PK_{mme}}$, (M8: Attach Accept): MME $\rightarrow$ eNodeB: $eNodeB \triangleleft \{\{MME \mid \Rightarrow KENB\}PK_{mme}^{-1}\}_{SK_{em}}$, (M9: AS Security Mode Command): eNodeB $\rightarrow$ UE: $UE \triangleleft \{eNodeB \mid \Rightarrow \{\#(RN_{enb}), \{AS-ALG-C\}PK_{enb}^{-1}\}\}_{PK_{ue}}$, (M10: AS Security Mode Complete): UE $\rightarrow$ eNodeB: $eNodeB \triangleleft \{\{UE \mid \Rightarrow ASDONE\}PK_{ue}^{-1}\}_{PK_{enb}}$.

The following initial Assumptions (A) are considered to evaluate the proposed protocol: **A1:** It is considered that some public keys in REPS-AKA3 (i.e. PK_{ue} , PK_{enb} , PK_{mme} and PK_{hss}) are used between involved entities as follows: $UE \models \xrightarrow{PK_{ue}}$
 $UE \wedge \xrightarrow{PK_{mme}} MME \wedge \xrightarrow{PK_{enb}} eNodeB$, $MME \models \xrightarrow{PK_{hss}} HSS \wedge \xrightarrow{PK_{ue}} UE \wedge \xrightarrow{PK_{mme}} MME \wedge \xrightarrow{PK_{enb}} eNodeB$,
 $HSS \models \xrightarrow{PK_{hss}} HSS \wedge \xrightarrow{PK_{mme}} MME$, $eNodeB \models \xrightarrow{PK_{ue}} UE \wedge \xrightarrow{PK_{mme}} MME \wedge \xrightarrow{PK_{enb}} eNodeB$, **A2:** It is

considered that some secret keys in REPS-AKA3 (i.e. SKmh, and SKem) are used between different nodes as follows: $MME \models eNodeB \xleftrightarrow{SKem} MME \wedge MME \xleftrightarrow{SKmh} HSS, HSS \models HSS \xleftrightarrow{SK} UE \wedge MME \xleftrightarrow{SKmh} HSS, eNodeB \models eNodeB \xleftrightarrow{SKem} MME, \mathbf{A3}$: It is assumed that the entities calculate (i.e., control) some security parameters and keys as follows: $UE \models HMSI \wedge UI \wedge HI \wedge NKG \wedge MAC \wedge RES \wedge CK \wedge IK \wedge AK \wedge KASME \wedge AUTN \wedge KNASenc \wedge KNASint \wedge KeNB \wedge NKeNB \wedge KRRCenc \wedge KRRCCint \wedge KRRCUPenc \wedge LAUTHP \wedge UK \wedge URAND \wedge UAUTN \wedge URES \wedge UKASME, MME \models HNID \wedge KNASenc \wedge KNASint \wedge KeNB \wedge LAUTHP \wedge UK \wedge URAND \wedge UAUTN \wedge UXRES \wedge UKASME, HSS \models HMSI \wedge HNID \wedge UI \wedge HI \wedge NKG \wedge MAC \wedge XRES \wedge CK \wedge IK \wedge AK \wedge KASME \wedge AUTN, eNodeB \models KeNB \wedge NKeNB \wedge KRRCenc \wedge KRRCCint \wedge KRRCUPenc, \mathbf{A4}$: We assumed that the entities generate some random value to use in key derivation and authentication functions as follows: $UE \models \#(RNue), HSS \models \#(RAND), \#(RNhss), MME \models \#(RNmme), eNodeB \models \#(RNenb), \mathbf{A5}$: It is considered that the entities have some security value related to other involved nodes as follows: $UE \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge (UE \xleftrightarrow{SQN} HSS) \wedge (UE \xleftrightarrow{NAS-CAD} MME) \wedge (UE \xleftrightarrow{NAS-IAD} MME) \wedge (UE \xleftrightarrow{NAS-ALG-ID} MME) \wedge (UE \xleftrightarrow{NAS-UP-C} MME) \wedge (UE \xleftrightarrow{NAS-DONE} MME) \wedge (UE \xleftrightarrow{NAS-ALG-C} MME) \wedge (UE \xleftrightarrow{UECap} MME) \wedge (UE \xleftrightarrow{AS-IAD} eNodeB) \wedge (UE \xleftrightarrow{AS-EAD} eNodeB) \wedge (UE \xleftrightarrow{AS-UPeAD} eNodeB) \wedge (UE \xleftrightarrow{AS-DONE} eNodeB) \wedge (UE \xleftrightarrow{AS-ALG-C} eNodeB) \wedge (UE \xleftrightarrow{AS-ALG-ID} eNodeB), MME \models (UE \xleftrightarrow{NAS-CAD} MME) \wedge (UE \xleftrightarrow{NAS-IAD} MME) \wedge (UE \xleftrightarrow{NAS-ALG-ID} MME) \wedge (UE \xleftrightarrow{NAS-UP-C} MME) \wedge (UE \xleftrightarrow{NAS-DONE} MME) \wedge (UE \xleftrightarrow{NAS-ALG-C} MME) \wedge (UE \xleftrightarrow{UECap} MME) \wedge (MME \xleftrightarrow{MVP} HSS) \wedge (MME \xleftrightarrow{HVP} HSS), HSS \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge (UE \xleftrightarrow{SQN} HSS) \wedge MME \xleftrightarrow{MVP} HSS) \wedge (MME \xleftrightarrow{HVP} HSS), eNodeB \models (UE \xleftrightarrow{AS-IAD} eNodeB) \wedge (UE \xleftrightarrow{AS-EAD} eNodeB) \wedge (UE \xleftrightarrow{AS-UPeAD} eNodeB) \wedge (UE \xleftrightarrow{AS-DONE} eNodeB) \wedge (UE \xleftrightarrow{AS-ALG-C} eNodeB) \wedge (UE \xleftrightarrow{AS-ALG-ID} eNodeB).$

The main security Goals (**G**) of the suggested protocol (REPS-AKA3) are to provide the mutual authentication between involved entities and manage security keys between user side and network side. All defined security goals in the suggested method are represented as follows: **G1**: The mutual authentication between involved nodes is

ensured in the suggested method as $UE \models HSS \models MME \wedge MME \models HSS \models UE$ then $UE \models MME \wedge MME \models UE$ and also as $UE \models MME \models eNodeB \wedge eNodeB \models MME \models UE$ then $UE \models eNodeB \wedge eNodeB \models UE$, **G2:** Key acceptance achieved between different defined entities. There is a matched KASME between the UE, HSS, and MME to run the proper key agreement. As $MME \models UE \models KASME \wedge UE \models MME \models KASME$ then $UE \xleftrightarrow{KASME} MME$, thus, as $UE \xleftrightarrow{KASME} HSS$ then $UE \xleftrightarrow{KNASenc} MME \wedge UE \xleftrightarrow{KNASint} MME$. Also, there is a matched KeNB key between the UE, and eNodeB to run the key agreement. These nodes can generate NKeNB by using KeNB. As $eNodeB \models UE \models NKeNB \wedge UE \models eNodeB \models NKeNB$ then $UE \xleftrightarrow{NKeNB} eNodeB$, thus, $UE \xleftrightarrow{NKeNB} eNodeB$, then $UE \xleftrightarrow{KRRCint} eNodeB \wedge UE \xleftrightarrow{KRRCenc} eNodeB \wedge UE \xleftrightarrow{KRRUPenc} eNodeB$, **G3:** Message authentication between corresponding entities as follows: $MME \models UE \sim HMSI$ (in M1: Attach Request), $HSS \models MME \sim HNID$ (in M2: Auth. Key Request), $MME \models HSS \sim HI$ (in M3: Auth. Key Reply), $UE \models MME \sim HI$ (in M4: UE Auth. Request), $MME \models UE \sim RES$ (in M5: UE Auth. Reply), $UE \models MME \sim NAS-ALG-C$ (in M6: NAS Security Mode Command), $MME \models UE \sim NASDONE$ (in M7: NAS Security Mode Complete), $eNodeB \models MME \sim KENB$ (in M8: Attach Accept), $UE \models eNodeB \sim AS-ALG-C$ (in M9: AS Security Mode Command), $eNodeB \models UE \sim ASDONE$ (in M10: AS Security Mode Complete), **G4:** Key agreement in AS and NAS layers are satisfied in involved nodes as follows: $UE \models HSS \models NKG \wedge HSS \models UE \models NKG$ then $UE \xleftrightarrow{NKG} HSS$ and also $UE \models MME \models UK \wedge MME \models UE \models UK$ then $UE \xleftrightarrow{UK} MME$ thus these involved nodes can generate other shared keys such as KeNB, NKeNB, and KASME, **G5:** The key freshness must be provided between UE and network side (i.e., MME, eNodeB and HSS) and also among network nodes. The key freshness between these entities holds since $UE \models HSS \models \{\#(RNhss), \#(RNue), \#(RNmme)\} \wedge HSS \models UE \models \{\#(RNhss), \#(RNue), \#(RNmme)\}$, thus, CK, IK, AK, UK, NKG, KeNB, and KASME are fresh keys accordingly, **G6:** Confidentiality between the UE (as a user side) and the MME (as a network side):

$$\frac{UE \models (UE \xrightarrow{KNASenc} MME), MME \triangleleft \{X\}_{KNASenc}}{UE \models MME \mid \sim \{X\}_{KNASenc}} \wedge \frac{MME \models (UE \xrightarrow{KNASenc} MME), UE \triangleleft \{Y\}_{KNASenc}}{MME \models UE \mid \sim \{Y\}_{KNASenc}} \text{ Where 'X' and 'Y'}$$

are transmitted messages from UE and MME respectively. Since we derived KNASenc using agreed KASME.

We evaluate the idealized form of presented method according to the hypothesis we defined and the BAN Logic rules including Message meaning rule (R1), Jurisdiction rule (R2), and Nonce Verification Rule(R3). We employ the BAN logic rule (R2) to

$$\text{derive: } \frac{UE \models HSS \mid \Rightarrow AUTN, UE \models HSS \mid \Rightarrow AUTN}{UE \models AUTN}, \frac{HSS \mid UE \mid \Rightarrow UI, HSS \mid \Rightarrow UE \mid \Rightarrow UI}{HSS \mid \Rightarrow UI}, \frac{UE \models HSS \mid \Rightarrow HI, UE \models HSS \mid \Rightarrow HI}{UE \models HI}$$

$$\frac{HSS \models MME \mid \Rightarrow HNID, HSS \models MME \mid \Rightarrow HNID}{HSS \models HNID}, \frac{eNodeB \models UE \mid \Rightarrow KeNB, eNodeB \models UE \mid \Rightarrow KeNB}{eNodeB \models KeNB},$$

$$\frac{MME \models UE \mid \Rightarrow RES, MME \models UE \mid \Rightarrow RES}{MME \models RES}, \frac{MME \models HSS \mid \Rightarrow XRES, MME \models HSS \mid \Rightarrow XRES}{MME \models XRES} \text{ To satisfy G1. We utilize the}$$

$$\text{BAN logic rule (R1) to derive: } \frac{MME \models (MME \xrightarrow{SKmh} HSS), MME \triangleleft \{KASME\}_{SKmh}}{MME \models HSS \mid \sim KASME} \wedge HSS \mid \Rightarrow$$

$$KASME, \frac{eNodeB \models (eNodeB \xrightarrow{SKem} MME), eNodeB \triangleleft \{KeNB\}_{SKem}}{eNodeB \models MME \mid \sim KeNB} \wedge MME \mid \Rightarrow KeNB \text{ To satisfy G2. We}$$

use the BAN logic rule (R1) to derive:

$$\frac{MME \models (PKue \xrightarrow{} UE), MME \triangleleft \{HIMSI\}_{PKue^{-1}}}{MME \models UE \mid \sim HIMSI}, \frac{HSS \models (PKmme \xrightarrow{} MME), HSS \triangleleft \{HNID\}_{PKmme^{-1}}}{HSS \models MME \mid \sim HNID},$$

$$\frac{MME \models (PKhss \xrightarrow{} HSS), MME \triangleleft \{HI\}_{PKhss^{-1}}}{MME \models HSS \mid \sim HI}, \frac{UE \models (PKmme \xrightarrow{} MME), UE \triangleleft \{HI\}_{PKmme^{-1}}}{UE \models MME \mid \sim HI},$$

$$\frac{MME \models (PKue \xrightarrow{} UE), MME \triangleleft \{RES\}_{PKue^{-1}}}{MME \models UE \mid \sim RES}, \frac{UE \models (PKmme \xrightarrow{} MME), UE \triangleleft \{NAS-ALG-C\}_{PKmme^{-1}}}{UE \models MME \mid \sim NAS-ALG-C},$$

$$\frac{MME \models (PKue \xrightarrow{} UE), MME \triangleleft \{NASDONE\}_{PKue^{-1}}}{MME \models UE \mid \sim NASDONE}, \frac{eNodeB \models (PKmme \xrightarrow{} MME), eNodeB \triangleleft \{KENB\}_{PKmme^{-1}}}{eNodeB \models MME \mid \sim KENB},$$

$$\frac{UE \models (PKenb \xrightarrow{} eNodeB), UE \triangleleft \{AS-ALG-C\}_{PKenb^{-1}}}{UE \models eNodeB \mid \sim AS-ALG-C}, \frac{eNodeB \models (PKue \xrightarrow{} UE), eNodeB \triangleleft \{ASDONE\}_{PKue^{-1}}}{eNodeB \models UE \mid \sim ASDONE}, \text{ To satisfy}$$

G3. We employ the BAN logic rule (R2) to derive:

$$\frac{UE \models HSS \mid \Rightarrow NKG, UE \models HSS \mid \Rightarrow NKG}{UE \models NKG}, \frac{HSS \models UE \mid \Rightarrow NKG, HSS \models UE \mid \Rightarrow NKG}{HSS \models NKG}, \frac{MME \models UE \mid \Rightarrow UK, MME \models UE \mid \Rightarrow UK}{MME \models UK},$$

$$\frac{UE \models MME \mid \Rightarrow UK, UE \models MME \mid \Rightarrow UK}{UE \models UK} \text{ To satisfy G4. And then all derived keys and parameters}$$

from NKG, UK and (X) RNK can be proved accordingly. We employ the BAN logic

$$\begin{aligned}
&\text{rule (R3) to derive: } \frac{UE \mid \equiv \#(RNhss), UE \mid \equiv HSS \mid \sim RNhss}{UE \mid \equiv HSS \mid \equiv RNhss} \wedge \frac{UE \mid \equiv HSS \mid \Rightarrow RNhss, UE \mid \equiv HSS \mid \equiv RNhss}{UE \mid \equiv RNhss}, \\
&\frac{UE \mid \equiv \#(RNmme), UE \mid \equiv MME \mid \sim RNmme}{UE \mid \equiv MME \mid \equiv RNmme} \wedge \frac{UE \mid \equiv MME \mid \Rightarrow RNmme, UE \mid \equiv MME \mid \equiv RNmme}{UE \mid \equiv RNmme} \quad \wedge \quad UE \mid \equiv \#(RNue), \\
&\frac{HSS \mid \equiv \#(RNue), HSS \mid \equiv UE \mid \sim RNue}{HSS \mid \equiv UE \mid \equiv RNue} \wedge \frac{HSS \mid \equiv UE \mid \Rightarrow RNue, HSS \mid \equiv UE \mid \equiv RNue}{HSS \mid \equiv RNue} \\
&, \frac{HSS \mid \equiv \#(RNmme), HSS \mid \equiv MME \mid \sim RNmme}{HSS \mid \equiv MME \mid \equiv RNmme} \wedge \frac{HSS \mid \equiv MME \mid \Rightarrow RNmme, HSS \mid \equiv MME \mid \equiv RNmme}{HSS \mid \equiv RNmme} \quad \wedge \quad HSS \mid \equiv
\end{aligned}$$

$\#(RNhss)$, Thus, all fundamental keys (i.e., CK, IK, AK, KASME and KeNB) and derived keys (KNASenc, KNASint, KRRCint, KRRCenc, and KRRUPenc) are generated using fresh numbers. Hence the protocol satisfies G5. Using proofs in G2 and G4, we found that $UE \mid \equiv HSS \mid \equiv KASME$ and $MME \mid \equiv HSS \mid \equiv KASME$. And by (R2):

$$\frac{MME \mid \equiv HSS \mid \Rightarrow KASME, MME \mid \equiv HSS \mid \equiv KASME}{MME \mid \equiv KASME} \quad \text{Then, we have } UE \xleftrightarrow{KASME} MME \text{ to satisfy G6.}$$

5.2.4 Proof of security requirements

This section illustrates some correctness proofs for the presented REPS-AKA3 protocol with its achievements. The evaluations are as follows:

Claim 1: The risk of incorrect attach is much less than other existing protocols.

Proof: The eNodeB broadcasts the information of all genuine eNodeBs using AAT. Thus, the UE selects the eNodeBs with the highest count number in FDT as a valid one. This method is strong against some attacks because the number of fake eNodeB (attackers) is smaller than valid eNodeB (genuine network) because of high-priced eNodeB. Moreover, the valid eNodeB is selected according to SL, CN, and SA in FDT. The SA is number of last successful attach which plays a significant role in the final decision. Thus, even if the signal level (SL) of fake eNodeB is higher than genuine eNodeB, the UE will choose the node with highest priority in FDT.

Claim 2: The REPS-AKA3 protects system against IMSI cracking attacks.

Proof: In the EPS-AKA method, the IMSI value is the main user's identity in the LTE network. Privacy issues happen since the LTE user can be tracked using captured IMSI. The REPS-AKA3 achieves strong privacy by employing dynamic UE identities. The protocol uses a new mechanism for temporary ID reallocation. Thus, it can provide user privacy considerably. In the REPS-AKA3 protocol, the UE hides the IMSI using HIMS and transmits the encryption of its identity (i.e., HIMS) by the public key of MME (PK_{mme}). It also uses UI as a secret identifier for the UE. Only the HSS and UE have a secret key (i.e., SK) and can find the UI of a given UE. Hence, the HSS can retrieve the user identity of genuine UE securely.

Claim 3: The proposed REPS-AKA3 provides strong mutual authentication.

Proof: The REPS-AKA3 method can provide mutual authentication between UE and HSS, between MME and UE, UE and eNodeB, and between HSS and MME (EPC entities). The UE can verify the HSS and MME using AUTH, HI, and RN_{ue}. The MME authenticates UE and HSS by RES and RN_{mme} appropriately. The HSS can authenticate MME and UE by verifying HNID and UI parameters effectively. Finally, the eNodeB checks authentication of UE by RN_{enb} in M10. Thus, full mutual authentication is obtained when the involved entities authenticate each other.

Claim 4: The proposed REPS-AKA3 achieves strong protection of EPS-AVs.

Proof: The transmission of EPS-AV (i.e., XRES, KASME, AUTN, and RAND) is protected by a secret key (SK_{mh}). The HSS must confirm the MME before having this vector. The HSS can authenticate MME by received HNID in message M2. As a result,

a fake MME (i.e., attacker) needs to know the honest MME's secret key to have this vector, which is impossible in the REPS-AKA3.

Claim 5: The proposed REPS-AKA3 protocol is resistant to MitM attacks.

Proof: In REPS-AKA3 protocol, the user's identity (i.e., IMSI) cannot be retrieved or modified by an adversary since it is hidden by HIMSI. And, the UI is generated by RNue and SK, which is shared only between UE and HSS. Identifier pair (HIMSI, UI) is protected by a strong key (i.e., PKmme and SKmh). Moreover, the other information, HI, HIND, and AVs, are not transmitted in a clear text. Thus, REPS-AKA3 is protected against MitM attacks.

Claim 6: In REPS-AKA3 protocol, the main keys are always fresh and secure.

Proof: The standard EPS-AKA uses a symmetric key (i.e., K) to perform AKA procedure. It generates subsequent CK, IK, AK, and KASME based on key K. The attacker can disclose this key and then extract all the subsequent keys. Therefore, standard EPS-AKA can not cover perfect forward secrecy. In the REPS-AKA3 method, the UE employs a new secret key (i.e., NKG). The NKG is generated by applying the NKGFunc to a generated random number (RNmme), HI value, and the original symmetric key (SK): $NKG = NKGFunc((RNmme \oplus HI), SK)$. The main goal of this new key is to protect the AKA process from replay attacks. Other symmetric keys (i.e., SKmh, SK, and SKem) protect communication between nodes. In addition, generated keys (i.e., KASME, KeNB, and the UK) are created to make them very difficult to be extracted. Moreover, calculated keys in AS and NAS layers are computed based on random values which make them hard to break. Thus, the security

keys in all the authentication processes are generated by fresh random values. This mechanism provides perfect forward secrecy.

Claim 7: The protocol ensures resistance to eavesdropping attacks.

Proof: In a standard AKA scheme, attackers can steal the messages in wireless channels. However, in the REPS-AKA3 protocol, attackers cannot eavesdrop on transmitted parameters. Therefore, the REPS-AKA3 is safe under this adversary model. Furthermore, all transmitted parameters are encrypted as shown in Table 21. Thus, confidentiality is satisfied completely in the REPS-AKA3 protocol.

Table 21: Asymmetric and symmetric keys

Messages	Keys	Encryption types	Messages	Keys	Encryption types
<i>M1 (UE → MME)</i>	PKmme	Asymmetric	<i>M8 (MME → eNB)</i>	SKem	Symmetric
<i>M2 (MME → HSS)</i>	SKmh	Symmetric	<i>M9 (eNB → UE)</i>	PKue	Asymmetric
<i>M3 (HSS → MME)</i>	SKmh	Symmetric	<i>M10 (UE → eNB)</i>	PKenb	Asymmetric
<i>M4 (MME → UE)</i>	PKue	Asymmetric	<i>Re-M1 (UE → MME)</i>	PKmme	Asymmetric
<i>M5 (UE → MME)</i>	PKmme	Asymmetric	<i>Re-M2 (MME → UE)</i>	PKue	Asymmetric
<i>M6 (MME → UE)</i>	PKue	Asymmetric	<i>Re-M3 (UE → MME)</i>	PKmme	Asymmetric
<i>M7 (UE → MME)</i>	PKmme	Asymmetric			

Claim 8: The REPS-AKA3 achieves optimization of the computational cost.

Proof: In the REPS-AKA3 protocol, the RAND, XRES, AUTN, and KASME computed in HSS can be used for the future. The MME and UE generate a UK key to update these parameters. These nodes have URAND, UXRES, UAUTN, and UKASME for n^{th} attach. Thus, there is no need to compute them again in both UE and the network side, which helps to decrease the keys generation overload. In addition, new keys in both AS and NAS layers are based on UKASME and UK. Thus, we have a considerable decrease in computational cost in n^{th} attach.

Claim 9: The REPS-AKA3 protocol is resistant to redirection attacks.

Proof: The protocol is resistant to redirection attacks because of its strong mutual authentication. To achieve strong mutual authentication, the REPS-AKA3 employs the challenge-response mechanism. The HSS authenticates the UE and the MME by checking the validity of the received UI and HNID. For verifying the HSS, similarly, the UE computes and checks the AUTN and HI. Then, the MME checks the value of RES and compares it with XRES to verify the user. Thus, the REPS-AKA3 approach can prevent the deployment of fake LTE nodes, which enables redirection attacks.

Claim 10: This method provides strong authentication between MME and HSS.

Proof: The MME and the HSS employ two security parameters (i.e., MVP and HVP) to authenticate each other in AKA procedure. Indeed, the MVP and HVP parameters are an agreement between the home and serving networks. The HSS and MME employ these parameters to generate HNID and authenticate each other accordingly. Thus, this method protects the MME from impersonation attacks even if an attacker discloses the SNID value. Since the HNID is encrypted using the secret key (SK_{mh}), the attacker must know the SK_{mh} key to extract the HNID parameter.

Claim 11: The protocol achieves optimization of the computational cost in UE.

Proof: In REPS-AKA3, the UE sets the UECap to specify its capability for encryption algorithms in data/control planes. Since the UEs have different specifications, they cannot support all algorithms efficiently which affects the system QoS level. Based on the UECap parameter, both UE and network (i.e., MME and eNodeB) sides set AS-

IAD, AS-EAD, AS-UPEAD, NAS-CAD, and NAS-IAD to generate KRRCCint, KRRCCenc, KRRCCUPenc, NAS-CAD, and NAS-IAD, respectively.

Claim 12: The protocol achieves message authentication.

Proof: The REPS-AKA3 protocol provides the message authentication in LTE nodes (i.e., UE, eNodeB, MME and HSS) using inverse public keys. In this manner, LTE nodes can be sure that the message is from genuine nodes. This reduces the probability of DoS/DDoS attacks in LTE systems. The MME can authentication the M1 message using $\{HIMSI\}_{inv}(PK_{ue})$. The HIMSI is authenticated by MME as built by genuine UE, which has PK_{ue} . Message authentication requirement is satisfied in other messages, as shown in Table 22.

Table 22: Message authentication in REPS-AKA3

Messages	Authentication Descriptions
<i>M1 (UE → MME)</i>	The HIMSI is authenticated by MME as built by UE
<i>M2 (MME → HSS)</i>	The HNID is authenticated by HSS as built by MME
<i>M3 (HSS → MME)</i>	The HI is authenticated by MME as built by HSS
<i>M4 (MME → UE)</i>	The HI is authenticated by UE as built by MME
<i>M5 (UE → MME)</i>	The RES is authenticated by MME as built by UE
<i>M6 (MME → UE)</i>	The NAS-ALG-C is authenticated by UE as built by MME
<i>M7 (UE → MME)</i>	The NASDone is authenticated by MME as built by UE
<i>M8 (MME → eNB)</i>	The KeNB is authenticated by eNB as built by MME
<i>M9 (eNB → UE)</i>	The AS-ALG-C is authenticated by UE as built by eNB
<i>M10 (UE → eNB)</i>	The ASDone is authenticated by eNB as built by UE
<i>Re-M1 (UE → MME)</i>	The LAUTHP is authenticated by MME as built by UE
<i>Re-M2 (MME → UE)</i>	The RNmme is authenticated by UE as built by MME
<i>Re-M3 (UE → MME)</i>	The URES is authenticated by MME as built by UE

Claim 13: The protocol ensures strong key generation in AS security context.

Proof: The RRC keys (i.e., KRRCCint, KRRCCenc, and KRRCCUPenc) in both UE and network sides (i.e., eNodeB) are calculated using fresh NKeNB. Thus, the RRC keys

are fresh, and the attacker cannot capture and use them for the next attach. Therefore, the security requirement in AS security context is satisfied appropriately.

Claim 14: The protocol ensures secure key generation in the NAS layer.

Proof: The NAS keys (KNASenc and KNASint) are computed using the KDF function and generated KASME or UKASME. The UE and network generate the KASME and UKASME based on NKG and UK, which are fresh keys. Thus, the NAS keys are always fresh and have an acceptable level of security.

Claim 15: The protocol protects the system against single-key dependency issue.

Proof: The proposed REPS-AKA3 employs dynamic keys (i.e., NKG, UK) instead of permanent ones. In this protocol, the NKG is based on the secret key SK, HI values, and RNmme. The NKG and UK are used to make other keys, and no one has them except genuine nodes (i.e., UE, MME, and HSS). Thus, the NKG and UK keys are always fresh keys, and they are generated for only one attach between UE and the network. Thus, for each new authentication, the UE and HSS employ a fresh key.

Claim 16: The protocol provides a perfect resistance to the Replay attack.

Proof: In the REPS-AKA3 method, attach request message contains a fresh timestamp (i.e., TS) and RNue. This pair (RNue, TS) is transmitted to MME securely. The MME can verify the request message using the TS value, and if the TS is a valid timestamp, then the MME accepts this AKA request; otherwise, MME ignores the received request. Thus, MME verifies the timestamps (i.e., TS) to ensure that the packet is fresh.

In our scheme, the MME drops stale packets, and they are not forwarded to the HSS. Moreover, random values (i.e., RNmme, RNhss, and RNenb) in other messages can provide suitable protection against replay attacks in LTE systems.

The security of the REPS-AKA3 protocol is compared with some other methods. The comparison results are shown in Table 23 (Satisfied: S and Unsatisfied: U). As it can be seen from the table, REPS-AKA3 has the highest total number of satisfied features compared with other protocols, which means that it is more secure. In addition, another comparison of the proposed REPS-AKA3 is illustrated in Table 24. Again, as it can be seen from the table, the REPS-AKA3 outperforms the other protocols as it has the highest total number of satisfied features. These comparisons allow us to position REPS-AKA3 in the list of secure AKA protocols.

Table 23: Evaluation of security features in AKA methods

AKA Security Features	EPS-AKA [67]	ECAKA [45]	EMTC-AKA [9]	E ² SEC [66]	M-LTE AKA [25]	REPS-AKA3
<i>Secure against IMSI Leak</i>	U	S	S	S	S	S
<i>Protects against MitM attack</i>	U	S	S	S	S	S
<i>Protects against DoS attack</i>	U	U	U	U	U	S
<i>Protects against redirection attack</i>	U	U	U	U	U	S
<i>Secure against impersonation attack</i>	U	S	S	S	S	S
<i>Secure against replay attack</i>	U	U	S	S	U	S
<i>Strong mutual authentication</i>	U	U	U	U	U	S
<i>Independency of Single secret key</i>	U	U	U	S	U	S
<i>Secure against HSS overloading</i>	U	S	S	U	S	S
<i>Total number of satisfied features</i>	0	4	5	5	4	9

Table 24: Comparison of security goals in AKA protocols

Security Goals	Ref. [51]	Ref. [52]	Ref. [53]	Ref. [54]	Ref. [71]	Ref. [56]	Ref. [57]	Ref. [74]	Ref. [62]	Ref. [63]	Ref. [65]	REPS-AKA3
SG1	Sym	Hybrid	Asym	Sym	Sym	Hybrid	Sym	Sym	Sym	Asym	Sym	Hybrid
SG2	S	U	U	U	S	U	S	S	S	U	S	S
SG3	U	S	U	U	U	S	U	U	U	S	S	S
SG4	U	U	U	S	S	S	S	U	U	U	U	S
SG5	U	S	S	S	S	S	S	U	U	U	U	S
SG6	S	S	S	S	S	S	S	S	S	S	S	S
SG7	S	S	S	S	U	S	U	U	U	U	U	S
SG8	U	U	S	U	S	S	U	U	U	U	U	S
SG9	U	U	S	S	S	U	U	U	U	U	U	S
SG10	U	U	U	U	U	U	U	U	U	U	U	S
Total number	3	4	5	5	6	6	4	2	2	2	3	10

In Table 24, symbols are defined as S: Satisfied; U: Unsatisfied; SG1: Type of cryptosystem; SG2: Accept the 3GPP architecture; SG3: Protection of user's privacy; SG4: Resistance to redirection attack; SG5: Protection against MitM attacks; SG6: Secure against Replay attacks; SG7: Preservation from impersonation attacks; SG8: Resistance to DoS attacks; SG9: provide KFS and KBS features; SG10: Solve single key issues.

5.3 Performance analysis of the proposed model: REPS-AKA3

This section presents the proposed protocol's performance evaluation and compares the REPS-AKA3 protocol with other AKA schemes based on bandwidth consumption, computational cost, signaling overhead, and storage overhead.

5.3.1 Bandwidth consumption (core traffic)

We use a basic mobility model to analyze the bandwidth consumption between MME and HSS. The traffic flow in this scenario is proportional to Length of registration cell (L), Mean Density of UE (MD), and user's speed (V). The direction of UE's movement can be distributed over $[0-2\pi]$ as shown in Table 25 [12, 24, 41].

Table 25: Assumption values

Parameters of mobility model	value
<i>Number of cells (Ar)</i>	100
<i>Length of registration area (L)</i>	25.3 km
<i>Mean Density of UE (MD)</i>	350 / km ²
<i>Average speed of UE (V)</i>	4.8 km/h

The request Rate (R) and Total Requests (TR) are defined by (5.1) and (5.2) [12, 24, 41]:

$$R = (MD \times V \times L) / \pi \quad (5.1)$$

$$TR = R \times Ar \quad (5.2)$$

Where Ar is the total number of registration areas. Bandwidth consumption (BW_{mh}) between EPC nodes is based on the number of bits used in the AKA messages. Thus, the BW_{mh} can be expressed as [12, 24, 41]: $BW_{mh} = \sum_{k=1}^{k=n} (MSG)_k \times TR$ (5.3)

Where n is the number of AKA messages transmitted among LTE nodes during the REPS-AKA3 procedure, and $(MSG)_k$ is the k^{th} message length in bits. Accordingly, we compare the BW_{mh} of the REPS-AKA3 protocol with other methods [12, 24, 41] as follows:

The BW_{mh} of EPS-AKA: In this method, the MME receives AVs from HSS node by forwarding two authentication messages. The MME sends its request for the AVs to the HSS using $MSG1$. In response, the HSS forwards n AVs to the MME using $MSG2$. Hence, the number of bits can be calculated as: $MSG1 = (IMSI) + (SNID) = 128 + 48 = 176 \text{ bits}$, $MSG2 = AV\{(XRES) + (AUTH) + (RAND) + (K_{ASME})\}_n = 544n \text{ bits}$, Consequently, using equation (5.3) with the parameters in Table 25 and if the number of AVs per request is 7 (i.e., $n=7$), the overall bandwidth consumption (BW_{mh}) between MME and HSS can be found as 1.49 Mbits/s.

The BW_{mh} of the Modified LTE-AKA (MLTE-AKA): Based on this AKA protocol, messages $MSG1$ and $MSG2$ are shared between MME and HSS. The MME sends $MSG1$, including Uid and SNID, to HSS. And the HSS forwards n copies of AVs to MME using $MSG2$: $MSG1 = 2 \times (Uid) + (SN_{ID}) = 176 \text{ bits}$, $MSG2 = AV\{(XRES) + (AUTH) + (IMSI) + (RAND) + (K_{ASME}) + (SK)\}_n = 800n \text{ bits}$, Therefore, using equation (5.3) with the parameters in Table 25, and if the number of AVs per request is 7 (i.e., $n=7$), the overall bandwidth consumption (BW_{mh}) for the modified LTE-AKA will be 2.17 Mbits/s.

The BW_{mh} of the REPS-AKA3: In the proposed REPS-AKA3, messages shared between MME and HSS for AKA method are: $MSG1$ and $MSG2$. The MME requests AKA keys and other related parameters from the HSS using $MSG1$. And the HSS forwards authentication parameters to the MME using $MSG2$: $MSG1 = (RN_{mme}) + (RN_{ue}) + (HNID) + (HIMSI) + (UI) = 128 + 128 + 128 + 128 + 128 = 640 \text{ bits}$, $MSG2 = (HI) + (RN_{hss}) + (RAND) + (AUTH) + (K_{ASME}) + (XRES) + (RN_{mme}) = 128 + 128 + 128 + 128 + 256 + 32 + 128 = 928 \text{ bits}$. Hence, using equation (5.3) with the parameters in Table 25, the overall bandwidth consumption (BW_{mh}) for the proposed REPS-AKA3 method is 0.58 Mbits/s.

The evaluation results show that the achieved improvements of the REPS-AKA3 over the standard AKA protocol and Modified LTE-AKA are 61% and 73% respectively (when $n=7$). Additionally, comparison between these AKA protocols is depicted in Figure 38 for different authentication vector sizes. The bandwidth consumption of the suggested protocol is fixed, however, there is a considerable increase in bandwidth consumption of other protocols when the number of authentication vectors increases. Hence, we conclude that the REPS-AKA3 achieves better bandwidth consumption for the core traffic compared to other AKA methods, including the EPS-AKA protocol.

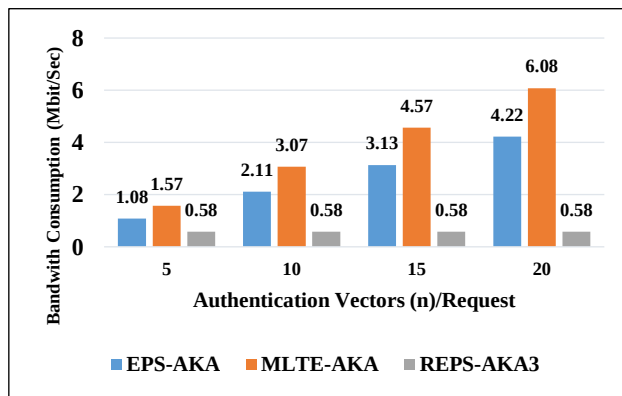


Figure 38: Bandwidth consumption (Core Traffic)

In addition, we evaluate the REPS-AKA3 and other methods based on MD . Figure 39 shows protocols behavior based on different MD . As expected, bandwidth consumption of all protocols increases as MD increases, however, REPS-AKA3 outperforms the others as it has the lowest bandwidth consumption.

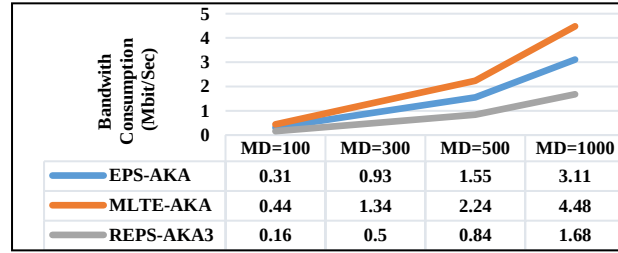


Figure 39: Bandwidth consumption of core traffic (AV=5)

5.3.2 Bandwidth consumption (total traffic)

We analyze the bandwidth consumption between involved nodes as shown in Figure 40 [12, 58]. The total bandwidth consumption of the proposed protocol is fixed. However, there is a considerable increase in the total bandwidth consumption of other protocols as the number of AV increases. The evaluation results show that the achieved average improvements of the REPS-AKA3 protocol over the EPS-AKA, Cocktail-AKA, and UMTS-AKA protocols are 56%, 57%, and 60% respectively.

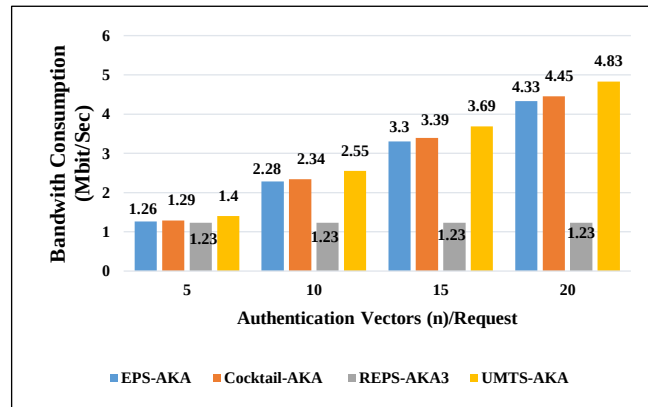


Figure 40: Total Bandwidth consumption

Since the number of UE plays a significant role in protocol performance, we compare the proposed protocol with other protocols according to the Mean Density (MD) of UE. Figure 41 shows that our protocol is much better than other protocols, especially when the MD increases. As expected, total bandwidth consumption of all protocols increases as MD increases, however, REPS-AKA3 outperforms the others as it has the lowest bandwidth consumption.

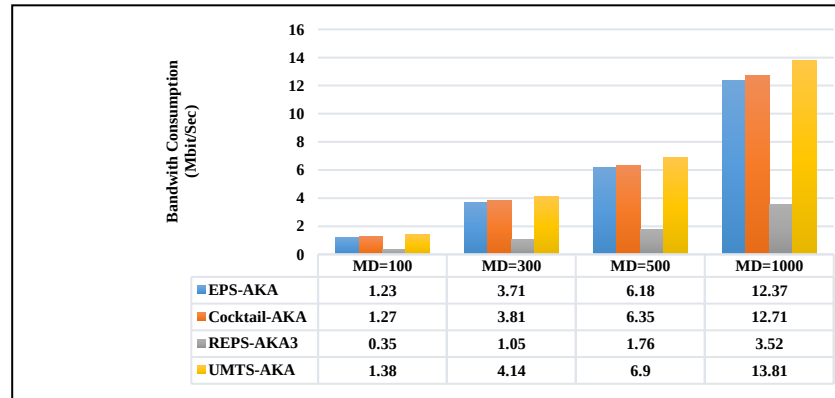


Figure 41: Total bandwidth in AKA protocols ($AV=20$)

In addition, we compare the total bandwidth consumption of EC-AKA [75], EPS-AKA [2], 3GPP's PKI [3,76], EC-AKA2 [74], and the proposed REPS-AKA3 protocols. Table 26 presents the transmitted bits of the above mentioned protocols. The total transmitted bits is divided into Upload Radio (UR), Upload Backhaul (UB), Download Radio (DR), Download Backhaul (DB), and Core Traffic (CT).

Table 26: Transmitted bits in AKA protocols

AKA Protocol	UR	UB	DR	DB	CT
EC-AKA	1172	1172	391	391	10240
EPS-AKA	160	160	304	304	5616
REPS-AKA3	944	944	768	768	1568
3GPP's PKI	1152	1152	1024	1024	9216
EC-AKA2	1172	1172	515	515	10260

Figure 42 illustrates the comparison of bandwidth consumption of the total traffic for the mentioned protocols. We assume that the number of transmitted AVs is 10 (i.e., $n=10$) and the TR is 376 according to the assumption values in Table 25.

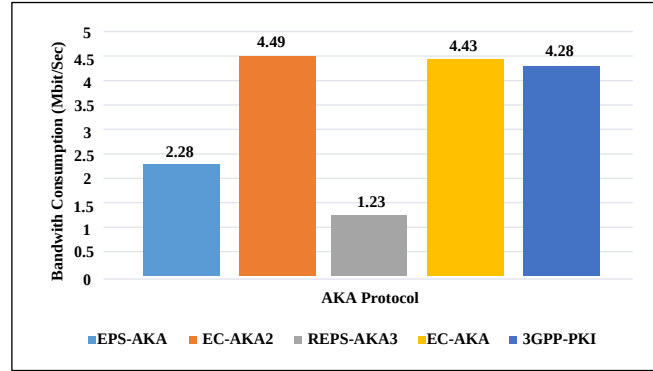


Figure 42: Total bandwidth consumption in AKA protocols

As can be seen in Figure 42, the REPS-AKA3 has outperformed other protocol in total bandwidth consumption. The evaluation results show that the achieved improvements of the REPS AKA3 protocol over the EPS AKA, EC AKA2, 3GPP PKI, and EC AKA are 46%, 72%, 71%, and 72% respectively.

Moreover, there are some methods which have better bandwidth consumption (e.g., Ref. [24]= $2032 \times TR$, Ref. [65]= $1890 \times TR$). But, according to other performance metrics and security evaluations, our method is better than these methods.

5.3.3 Computational cost

To evaluate the computational cost, we assume that each function's computation cost is 1 for all the security functions. Computational cost is calculated for the UE, MME, and the HSS based on the security functions based on (5.4) [12, 24, 41]:

$$Computation\ Cost = \sum_{k=1}^{k=m} Cost(node)_k \times TR \quad (5.4)$$

Where m is the number of nodes in the system (here, nodes are UE, MME, and HSS).

The EPS-AKA computational cost: In the standard EPS-AKA protocol, the computational cost of security functions performed by involved nodes (i.e., UE, MME, and HSS) are as follows: $UE_{cost} = (f1 + f2 + f3 + f4 + f5 + 2KDF) = 7$, $MME_{cost} = (No\ security\ function) = 0$, $HSS_{cost} = AV(f1 + f2 + f3 + f4 + f5 + 2KDF)n = 7n$, Where n is the number of AVs; hence, using equation (5.4), the total computation cost of the EPS-AKA can be calculated as: $EPS - AKA_{cost} = TR \times (7 + 7n)$.

The Modified LTE-AKA (MLTE-AKA) computational cost: Computational costs of security functions in involved nodes (i.e., UE, MME, and HSS) for performing Modified LTE-AKA procedure are as follows: $UE_{cost} = (Enc + 2fk + MAC + f1 + f2 + f3 + f4 + f5 + 3KDF) = 12$, $MME_{cost} = (No\ security\ function) = 0$, $HSS_{cost} = Dec + AV(3KDF + f1 + f2 + f3 + f4 + f5)n = 1 + 8n$, Where n is the number of AVs. Consequently, using equation (5.4), the total computation cost of the Modified LTE-AKA is: $Modified\ LTE - AKA_{cost} = TR \times (13 + 8n)$.

The REPS-AKA3 computational cost: Computational cost of security functions in the REPS-AKA3 protocol are as follows: $UE_{cost} = (HFunc + UIGFunc + HIGFunc + NKGFunc + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc) = 11$, $MME_{cost} = (NIFunc) = 1$, $HSS_{cost} = (HFunc + NIFunc + UIGFunc + HIGFunc + NKGFunc + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc) = 12$, Hence, using equation (5.4) the overall computation cost of the REPS-AKA protocol can be found as: $REPS - AKA_{cost} = TR \times (24)$.

Figure 36 shows the overall computational cost of various protocols with different numbers of AVs. We assume that TR is 376 (i.e., $MD=350$) according to the assumption values in Table 25. The results in Figure 43 show that the proposed REPS-AKA3 has less computational cost than other protocols. The computational cost of the

proposed protocol is fixed, however, there is a considerable increase in computational cost of the other protocols when the number of authentication vectors increases. The average improvements of REPS-AKA3 over EPS-AKA and Modified LTE-AKA are 78% and 74% respectively. The superiority of our protocol becomes more clear when the number of AV increases as shown in Figure 43.

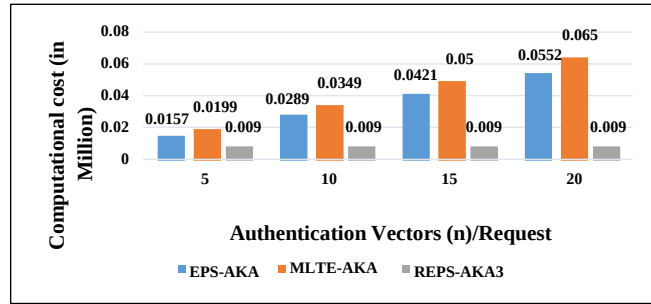


Figure 43: Computational cost analysis of AKA protocols

The comparative studies of the computational cost in different AKA protocols are shown in Figure 44. We evaluate AKA protocols based on MD . As can be seen, computational cost of all protocols increases as MD increases. But, REPS-AKA3 outperforms the others as it has the lowest computational cost and the smallest slope.

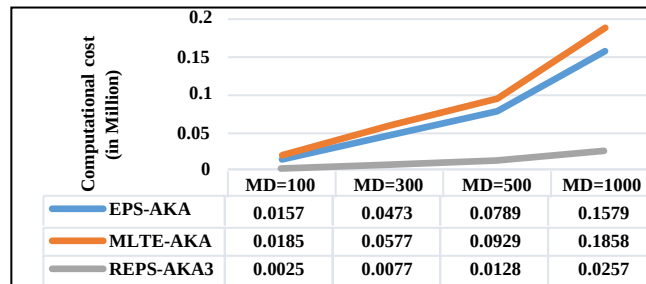


Figure 44: Computational cost in AKA protocols (AV=20)

5.3.4 Signaling overhead

All the AKA steps in REPS AKA3 method are identical to the steps in the standard EPS-AKA method. Thus, we have no extra round-trip delays in the proposed method

compared to the standard EPS-AKA. Thus, effectively, the REPS-AKA3 protocol does not generate any extra signaling overhead compared to the Standard AKA protocol. Table 27 illustrates the signaling overload of several AKA protocols [42] [53] [56] [54] [59][1] [71] [57] [51] [52] [58] [60].

Table 27: Comparison of signaling overhead

AKA protocol	Number of Message	AKA protocol	Number of Message
<i>G-AKA</i>	8	<i>GLARM-AKA</i>	8
<i>SE-AKA</i>	8	<i>PRIVACY-AKA</i>	8
<i>EG-AKA</i>	12	<i>GR-AKA</i>	7
<i>NOVEL-AKA</i>	8	<i>SEGB-AKA</i>	8
<i>GBAAM-AKA</i>	10	<i>REPS-AKA3</i>	6
<i>GROUP-AKA</i>	9	<i>EPS-AKA</i>	6

5.3.5 Storage overhead in the serving network

We evaluate the storage overhead of different methods in the MME. During AKA procedure, the MME stores AVs generated by the HSS for the next authentications. In the REPS-AKA3 method, the MME employs its UK key to compute new AVs. Thus, the MME keeps only one AV during the AKA procedure. The comparison of the storage overhead in some AKA protocols is shown in Table 28. Comparative results indicate that the REPS-AKA3 protocol can reduce the storage overhead.

Table 28: Storage overhead of AKA methods

AKA protocols	Storage Overhead (bits)
<i>Ref. [24]</i>	640n
<i>Ref. [61]</i>	448n
<i>Ref. [29]</i>	432n
<i>Ref. [10]</i>	392n
<i>Ref. [62]</i>	608n
<i>Ref. [63]</i>	704n
<i>Ref. [25]</i>	800n
<i>REPS-AKA3</i>	544

Storage overheads at MME for various suggested protocols are depicted in Figure 45. According to these results, we can conclude that REPS-AKA3 has the lowest storage overhead among other protocols. In addition, the storage overhead of the proposed protocol is fixed, however, there is a considerable increase in storage overhead of other methods when the number of authentication vectors increases.

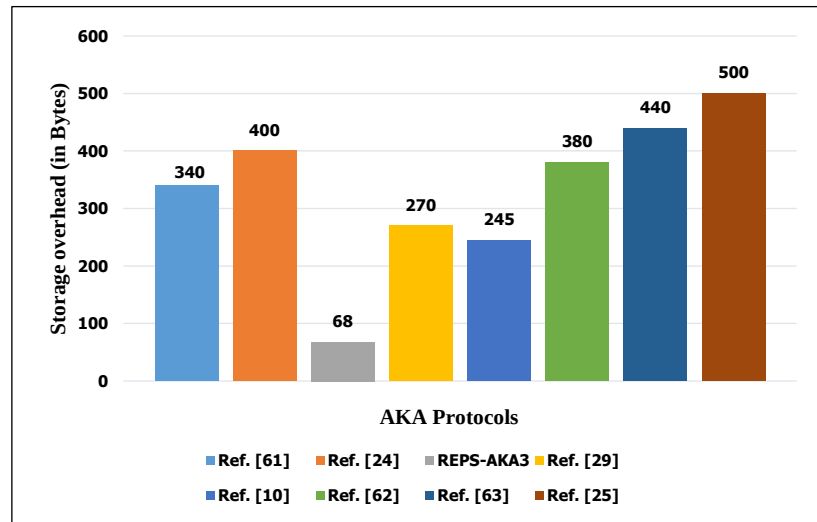


Figure 45: Storage overhead in AKA protocols (AV=5)

Chapter 6

MODEL 4 (REPS-AKA4)

There is a considerable demand for high performance networks with secure and reliable services in wireless systems. New generation of communication systems can provide this increasing demand properly by the interworking technology. The 3GPP network (e.g., LTE) can interworks with non-3GPP Network (e.g., WLAN) to improve mobility of users in wireless systems. Interworking needs an appropriate mobility management system to improve the level of accessibility in the involved networks. It also requires a proper AKA mechanism to provide uninterrupted and proper network services especially when users move between the different heterogeneous networks (e.g., between LTE and WLAN) [80] [81]. In Heterogeneous Networks (HetNets), the LTE merges the mobile communication system and WLAN to increase the coverage area and improve service of mobile networks. One of the most common interworking technologies is the LTE-WLAN Interworking (LWI) system. The LWI system provides suitable privileges to both user side and network side such as high QoS and lower cost. In addition, the number of smartphones supporting both LTE and WLAN technologies are growing rapidly. Thus, LWI system has become a practical environment for mobile users. This is due to the mobile capabilities, good QoS and wide coverage area of the LTE on one hand, and the high bit rate of the WLAN network on the other hand [82].

Integration of these heterogeneous wireless networks (i.e., LTE and WLAN) creates new security issues for interworking architecture especially in handover procedures. In the LWI architecture, authentication and re-authentication phases are the most significant stages to provide a safe connection during handover procedure [83] [84]. There are some suggested methods for handover procedure between different networks [84-93]. Some handover methods such as IEEE 802.11r, and IEEE 802.11i [85] are proposed to improve AKA delay in handover procedure. However, the mentioned methods just work in WLANs and are not applicable for LTE networks. The authors in [87] presented a new protocol to provide authentication between server and users in communication systems. However, the suggested method is not suitable for re-authentication phases. Mostafa et al defined an AKA method in [84] to provide suitable security level in WLAN-LTE systems. Also, there are some other proposed security protocols for 3G-WLAN networks [90-92]. Most of the suggested AKA methods are vulnerable to malicious threats, such as MiTM, and DoS attacks. The main challenge of LWI system is the authentication delay and signaling cost of AKA process between user and network side. Thus, we need an efficient AKA protocol to provide secure and seamless handover process in the LWI systems.

In this chapter, a set of REPS-AKA4 protocols are introduced as a robust inter/intra-authentication and re-authentication security mechanisms for the LWI systems. The security details of the proposed REPS-AKA4 protocols are presented and compared with other approaches in 6 different scenarios. The suggested approaches are proposed to preserve the confidentiality of the WLAN by using the LTE network address. The proposed REPS-AKA4 protocols can optimize the authentication delay, communication cost and simultaneously provide an acceptable security level in LWI

systems. The rest of this chapter is organized as follows: In section 6.1, we provide the overview of LWI system. In section 6.2, we present the suggested REPS-AKA4 methods to solve security issues of existing approaches. Section 6.3 presents a security evaluation of the REPS-AKA4 methods. Section 6.4 illustrates the performance analysis of the REPS-AKA4 protocols and compares them with the standard methods.

6.1 LTE-WLAN systems

The integration of the LTE with WLAN makes a high-grade technology for mobile users in the wireless networks because of their outstanding features. Important properties of LTE networks are reduced latency, improved network capacity, and high data rate [12,82]. The architecture of LWI network is shown in Figure 46. The WLAN Authentication Authorization Accounting (WAAA) and LTE AAA (LAAA) servers work as the bridge entities between WLAN and LTE domains. This connection is implemented by using two interfaces named as STa or Wa [82]. In a WLAN domain, mobile users connect to WLAN using Access Points (APs). Thus, we have two types of access points in LWI system: (1) the WLAN Access Point (AP) and eNodeB in LTE domains.

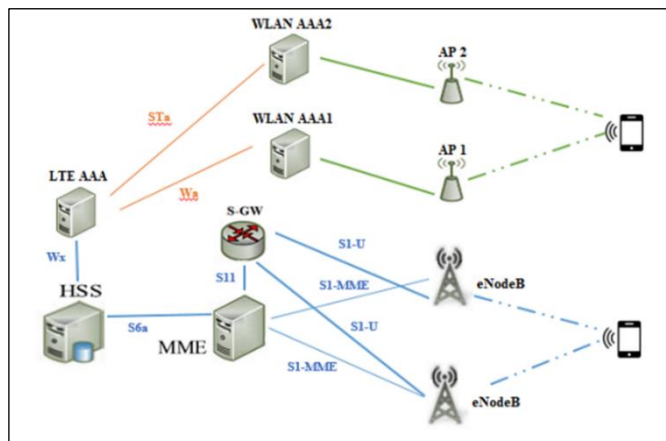


Figure 46: LWI architecture

6.2 The proposed REPS-AKA4 protocol

The standard protocols (i.e., LTE-AKA, Full EAP-AKA', and Fast EAP-AKA') have some limitations and security vulnerability [82]. Thus, we define the REPS-AKA4 protocols as an effective set of authentication and re-authentication methods for the LWI system. When user moves between different LTE and WLAN domains (i.e., 1, 2, 3, and 4) as shown in Figure 47, the UE employs the corresponding protocol.

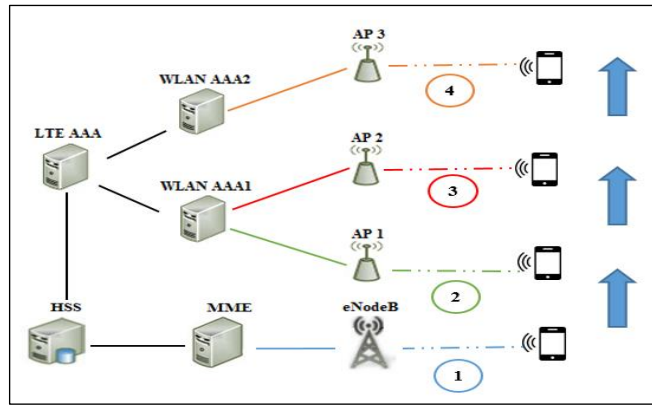


Figure 47: UE movements in LWI system

In domain 1, the user (i.e., UE) connects to the LTE network via eNodeB. In domain 2, the UE moves from the LTE network to the WLAN and connects to AP1. In domain 3, the UE moves between different WLANs and connects to AP2, (where AP1 and AP2 are in the same WLAN AAA1). In domain 4, the UE moves to another WLAN and connects to AP3, (where AP2 and AP3 are connected to different WLAN servers).

In Figure 48, we present a general diagram which shows how a UE selects the adequate AKA protocol based on its situation in LWI network.

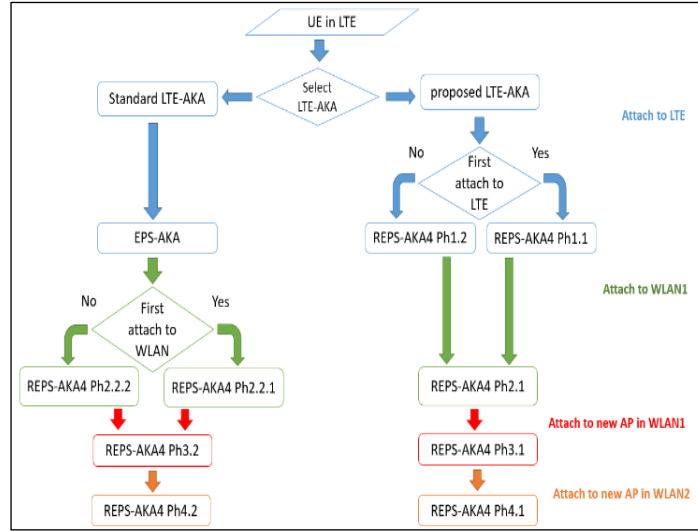


Figure 48: General diagram of UE selections for AKA protocols

The proposed approach is divided into 9 separate protocols. In Figure 49, the procedure of AKA protocol (i.e., REPS-AKA4) selection is presented in general.

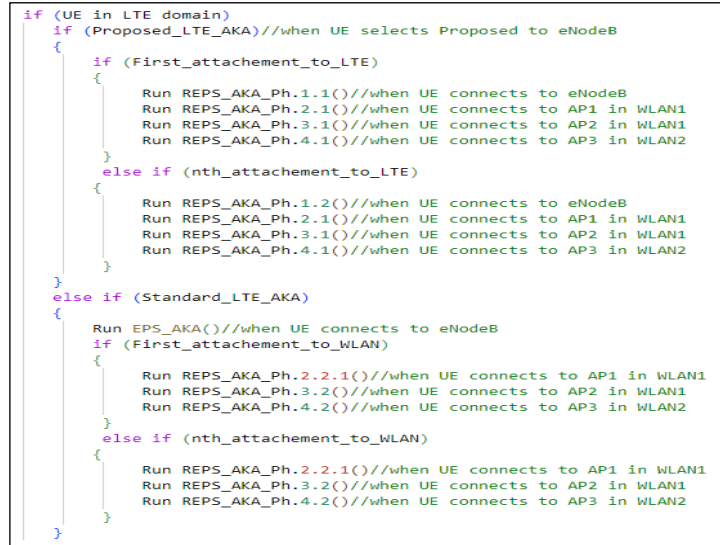


Figure 49: Procedure of AKA selection

6.2.1 REPS-AKA4 Ph1.1

The steps of the REPS-AKA4 Ph1.1 are presented in the following steps (Figure 50):

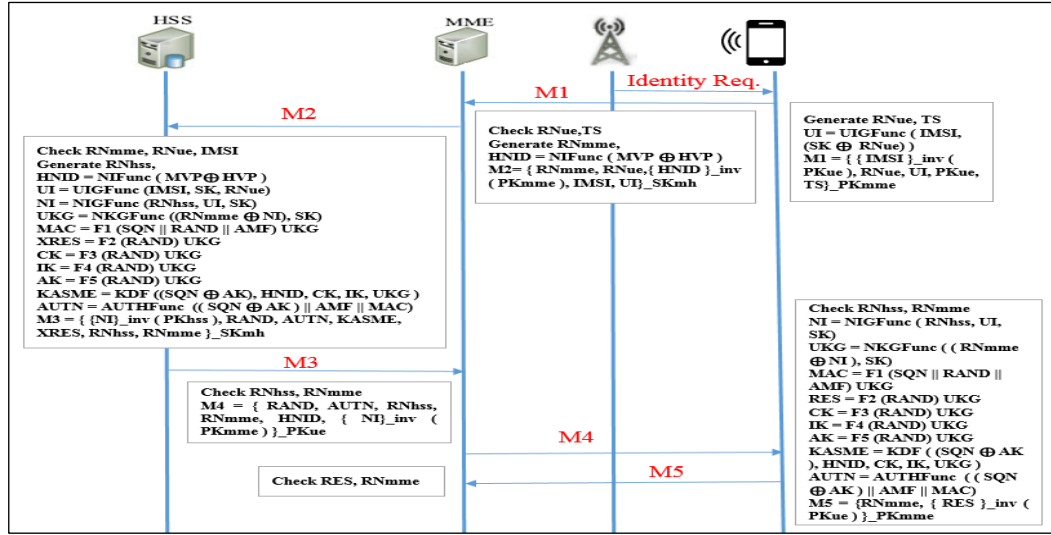


Figure 50: Steps of REPS-AKA4 Ph1.1

1. UE → MME (M1): When a UE connects to an LTE system for the first attachment, it sends its identity via M1 message. First, the UE produces an RNue (which is a random number) and a valid timestamp (TS). The MME uses this pair (i.e., RNue and TS) to confirm that the received message is fresh and valid. Then, the UE employs the UIGFunc function and the IMSI value to generate User Identity (UI) value as a unique identifier between HSS (as a network side) and UE (as a user side): $UI = UIGFunc(IMSI, (SK \oplus RNue))$. Then, the UE uses its inverse public key (PKue) to encrypt the IMSI value. Consequently, the UE transmits M1 message including $\{IMSI\}_{inv}(PKue)$, RNue, UI, public key of UE (PKue), and TS. The UE encrypts message M1 using MME public key (i.e., PKmme).

2. MME → HSS (M2): In the serving network, the MME decrypts the received message (i.e., M1) using its private key. First, the MME checks the freshness of RNue and TS in message M1. Then, it generates a random number (i.e., RNmme) and makes the HNID parameter which is Home Network Identity. This parameter is a unique identifier between the serving network (i.e., MME) and the home network (i.e., HSS).

The MME calculates HNID using pre-shared values (i.e., MVP and HVP): $HNID = NIFunc(MVP \oplus HVP)$. The MME uses its inverse public key to encrypt the HNID parameter. Then, the MME generates message M2 which contains RNmme, RNue, $\{HNID\}_{inv}(PK_{mme})$, IMSI, and UI. Finally, the MME encrypts message M2 by SKmh (secret key between MME and HSS) and forwards it to HSS.

3. HSS $\rightarrow$ MME (M3): The HSS decrypts the received message (i.e., M2) to extract RNmme, RNue, HNID, IMSI, and UI. The HSS calculates HNID and compares it with the received one to authenticate the serving network (i.e., MME). When MME is successfully authenticated, the HSS generates UI value using UIGFunc function, IMSI, and RNue, SK (which is pre-shared key): $UI = UIGFunc(IMSI, SK, RNue)$. Then, the HSS compares the calculated UI with the received one to authenticate UE. Thus, the HSS verifies the UE (as a valid user) and the MME (as a valid serving network) by checking UI and HNID values, respectively. If the UE and MME are verified successfully, then HSS generates its random number (RNhss) and Network Identifier (NI): $NI = NIGFunc(RNhss, UI, SK)$. The HSS encrypts the NI value using its inverse public key (PKhss) and forwards it to the MME via message M3. Moreover, the HSS calculates Updated Key Generator (UKG): $UKG = NKGFunc((RNmme \oplus NI), SK)$. Then, the HSS generates a RAND value which is a new random number. It uses the SQN, RAND, UKG, AMF, and HNID to generate $MAC = F1(SQN || RAND || AMF)UKG$, $AK = F5(RAND)UKG$, $CK = F3(RAND)UKG$, $XRES = F2(RAND)UKG$, $AUTN = AUTHFunc((SQN \oplus AK) || AMF || MAC)$, $IK = F4(RAND)UKG$, and $KASME = KDF((SQN \oplus AK).HNID.CK.IK)$. The HSS encrypts M3 message by SKmh and forwards it to MME. The message M3 includes $\{NI\}_{inv}(PK_{hss})$, RAND, AUTN, KASME, XRES, RNhss, and RNmme.

4. MME → UE (M4): The MME employs SK_{mh} to decrypt the received message. Then, it sends message M4 to the UE. This message which is encrypted by PK_{ue} includes RAND, AUTN, RN_{hss}, RN_{mme}, HNID, and {NI}_inv (PK_{mme}).

5. UE → MME (M5): The UE decrypts the M4 and checks the received random values (i.e., RN_{hss} and RN_{mme}) for freshness. Then, the UE produces NI value by received RN_{hss}, UI, and SK: $NI = NIGFunc(RN_{hss}, UI, SK)$. Then it compares the generated NI with the received NI to authenticate HSS. When the HSS is authenticated successfully, the UE generates the UKG key: $UKG = NKGFunc((RN_{mme} \oplus NI), SK)$. The UE calculates the value of MAC, AUTN, AK, CK, KASME, IK, and RES same as HSS side. The UE uses the AUTN to authenticate the network. Then, it employs KASME as a master key in AS and NAS layers. The UE forwards RN_{mme}, and {RES}_inv (PK_{ue}) to the MME via M5 encrypted by PK_{mme}.

6. MME: The MME decrypts M5 and extracts RES value. First, it verifies freshness of received message by checking RN_{mme}. Then, it compares the received RES with saved XRES to confirm the UE. If this match is successful, the AKA is successful.

6.2.2 REPS-AKA4 Ph1.2

When a UE stays in an LTE domain (under a given MME domain) for a long duration and wants to connect the network for the n^{th} time, it sends the n^{th} attach request to the LTE network. In this case, the UE and MME can use their previously generated parameters for AKA process. Thus, if a user wants a re-attach to the LTE network, Re-authentication phase of the proposed protocol starts to execute as follows (Figure 51):

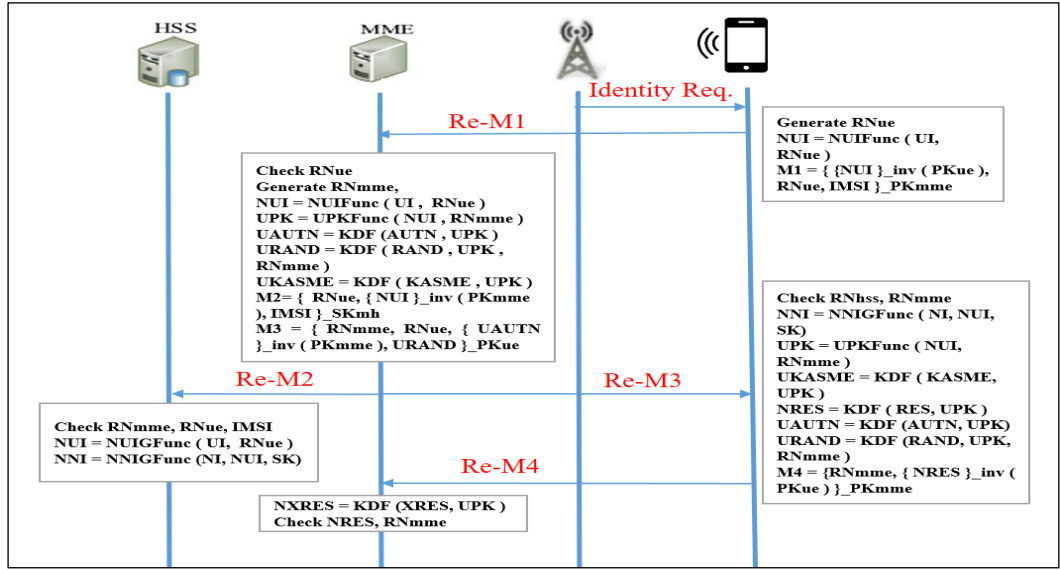


Figure 51: Steps of REPS-AKA4 Ph1.2

1. UE → MME (Re-M1): The UE generates a new random number (i.e., RNue). Then, it employs NUIFunc, UI and RNue to generate New User Identity (NUI): $NUI = NUIFunc (UI, RNue)$. The UE forwards NUI value to the MME to prove that it is previously connected to this MME. It sends Re-M1 encrypted by PKmme to the MME. The message Re-M1 includes $\{NUI\}_{inv} (PKue)$, RNue, and IMSI.

2. MME → UE (Re-M3)/HSS (Re-M2): The serving MME uses its private key to decrypt the received Re-M1. The MME employs RNue and UI to generate NUI. Then, it compares calculated NUI with received NUI to authenticate the UE (i.e., LTE user). The MME uses NUI, RNmme to calculate Updated Key (UPK): $UPK = UPKFunc (NUI, RNmme)$. Then, the MME generates an Updated AUTN (UAUTN) and an Updated RAND (URAND) and an Updated KASME (UKASME): $UAUTN = KDF(AUTN, UPK)$, $URAND = KDF (RAND, UPK, RNmme)$, $UKASME = KDF(KASME, UPK)$. The MME sends Re-M2 to HSS encrypted by secret key (i.e., SKmh) including RNue, $\{NUI\}_{inv}(PKmme)$, and IMSI, and forwards Re-

M3 to UE encrypted by public key of UE (i.e., PKue). The Re-M3 contains RNmme, RNue, {UAUTN}_inv (PKmme), and URAND.

3. HSS: The HSS decrypts Re-M2 and checks NUI value. It generates NUI using received RNue, and UI: $NUI = NUIFunc(UI, RNue)$. The HSS utilizes the NI, NUI, and SK to calculate New Network Identity (NNI): $NNI = NNIGFunc(NI, NUI, SK)$.

4. UE → MME (Re-M4): The UE decrypts Re-M3 and checks RNmme for freshness and generates NNI using NI, NUI, and SK: $NNI = NNIGFunc(NI, NUI, SK)$. The UE makes UPK: $UPK = UPKFunc(NUI, RNmme)$ and utilizes it to generate an UAUTN and an URAND and an UKASME same as MME. The UE makes New RES (NRES): $NRES = KDF(RES, UPK)$. Then, it forwards RNmme, and {NRES}_inv(PKue) to MME via message Re-M4 which is encrypted by PKmme.

5. MME: The MME receives message Re-M4 and extracts RNmme and NRES. Then, MME generates NXRES and compares it with NRES to authenticate the UE.

6.2.3 REPS-AKA4 Ph2.1

When the UE move to WLAN domain, the LTE network can provide IP services for the UE via the WLAN. In this state, the UE connects to WLAN using AKA information in LTE network. This information is located in HSS and LAAA server. The UE employs REPS-AKA4 Ph2.1 protocol when it is in LTE network before and it connects to WLAN using AKA parameter from that LTE network. The steps of this phase of the REPS-AKA4 are as follows (Figure 52):

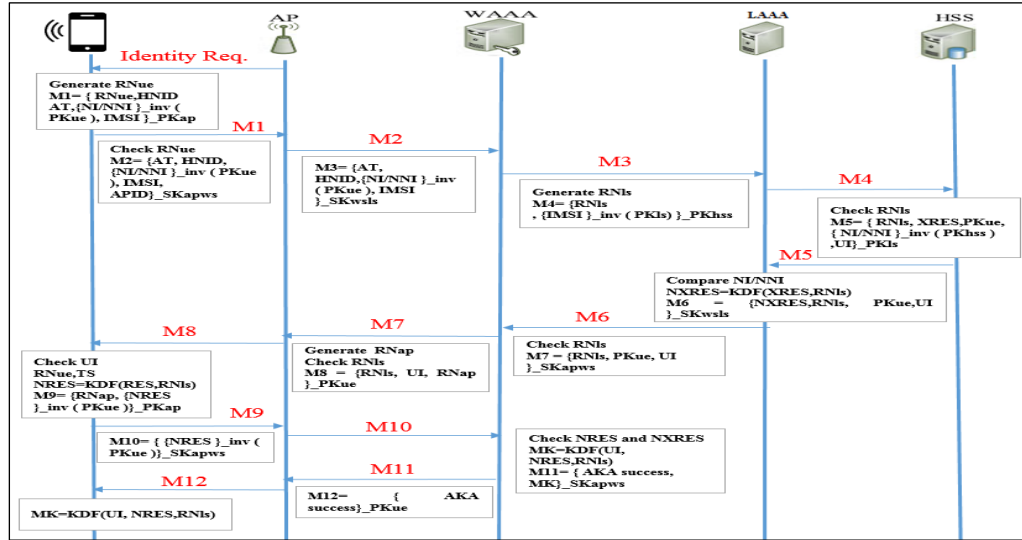


Figure 52: Steps of REPS-AKA4 Ph2.1

1. UE → AP (M1): When an UE tries to connect to WLAN-LTE network, and has previous identifier in LTE access network, it has to identify itself by transmitting a valid identifier. The UE generates a RNue. Then, it sets Authentication Type (AT) field to 1. The WLAN-LTE network use this field to figure out that it must use REPS-AKA4 Ph2.1 protocol for AKA procedure. The UE set Home Network Identifier (HNID) to inform network that it authenticated by which network before. It also sends NI or NNI value, encrypted by its inverse public key (PKue): {NI/NNI}_inv (PKue). Then, the UE sends M1 including RNue, HNID AT, {NI/NNI}_inv(PKue), and IMSI. The M1 is encrypted by public key of AP (PKap).

2. AP → WAAA (M2): The AP decrypts M1 and checks RNue for freshness. The AP uses the Access Point ID (APID) as a valid ID for WAAA server. Then, it sends M2 including AT, HNID, {NI/NNI}_inv(PKue), IMSI, and APID. The M2 is encrypted by pre-shared key (SKapws).

3. WAAA → LAAA (M3): The WAAA decrypts M2 with its SKapws. The WAAA server forwards the M3 to LAAA server. The information in M3 are AT, HNID, {NI/NNI}_inv (PKue), and IMSI. The M3 is encrypted by pre-shared key (SKwsls).

4. LAAA → HSS (M4): The LAAA decrypts M3 and generates a random number (RNls). Then, it forwards the M4 to HSS in LTE network. The information in M4 are RNls, and {IMSI}_inv(PKls). The M4 is encrypted by HSS public key (PKhss).

5. HSS → LAAA (M5): The HSS checks RNls for freshness and finds the information of the user according to IMSI value. These parameters are RNls, XRES, PKue, NI/NNI, and UI which are sent to LAAA server by M5 encrypted by PKhss.

6. LAAA → WAAA (M6): When LAAA received the M5, it compares saved NI/NNI with received NI/NNI makes a New XRES (NXRES) using XRES (i.e., previous one) and RNls: $NXRES = KDF(XRES, RNls)$. Then, it sets M6 including NXRES, RNls, PKue, and UI. The message M6 is encrypted by SKswls.

7. WAAA → AP (M7): When the WAAA server receives the M6, it checks RNls and sends M7 to AP in WLAN. The M7 including RNls, PKue, and UI. The M7 is encrypted by SKapws.

8. AP → UE (M8): The AP uses SKapws key to decrypt the received messages from WAAA and gets the values of RNls, PKue, and UI. Then, it generates Random Number (RNap) and checks RNls for freshness. The AP forwards RNls, UI, and RNap to UE via M8 encrypted by PKue.

9. UE → AP (M9): The UE checks the received UI to confirm network and makes NRES using pervious RES and RNls: $NRES = KDF(RES, RNls)$. Then, the UE forwards RNap, and $\{NRES\}_{inv}(PKue)$ to AP in M9 encrypted by P_{Kap}.

10. AP → WAAA (M10): The AP forwards $\{NRES\}_{inv}(PKue)$ to WAAA server via M10 encrypted by S_{Kapws}.

11. WAAA → AP (M11): The WAAA server checks the received NRES to confirm UE and makes Master Key (MK): $MK = KDF(UI, NRES, RNls)$. Then, it sends M11 to AP including AKA success, and MK. This message is encrypted by S_{Kapws}.

12. AP → UE (M12): The AP saves MK and forwards “AKA success” to UE via M12 encrypted by P_{Kue}.

13. UE: The UE checks the “AKA success” in received M12 and makes MK: $MK = KDF(UI, NRES, RNls)$.

6.2.4 REPS-AKA4 Ph2.2.1

This protocol has been specified to provide authentication when the UE did not connect to the LTE network before but its security information is saved in HSS and LAAA server. This method is for the situation where the UE connects to the AP of LWI for the first time. The steps of this method are presented as follows (Figure 53):

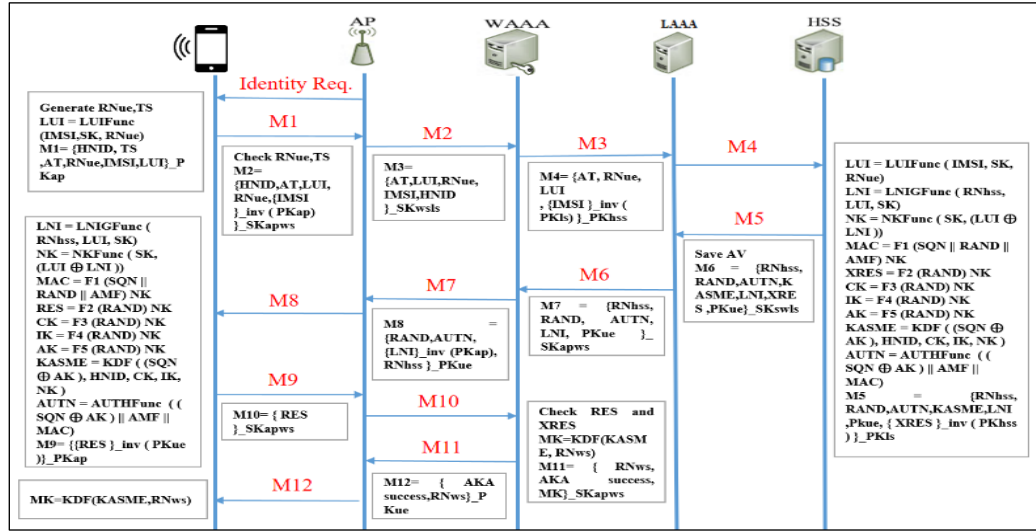


Figure 53: Steps of REPS-AKA4 Ph2.2.1

1. UE → AP (M1): When a user (i.e., UE) tries to connect to LWI system, but it has no previous identifier in LTE network, it has to identify itself by transmitting a valid identifier via M1 message. The UE generates an RNue which is a new random number and sets TS which is a valid timestamp. The AP uses this pair (i.e., RNue and TS) to confirm that the received request message is fresh. Then, it makes LTE User Identity (LUI): $LUI = LUIFunc(IMSI, SK, RNue)$ and sets AT field to 2. The LTE-WLAN network use this filed to figure out that it must use REPS-AKA4 Ph2.2.1 protocol for AKA process. The UE uses HNID to inform the LWI system that it is belong to which LTE network. Consequently, the UE transmits M1 including HNID, TS, AT, RNue, IMSI, and LUI. The message M1 is encrypted by P_{Kap}.

2. AP → WAAA (M2): The AP uses its private key to decrypt message M1. Then it checks RNue and TS for message freshness. The AP forwards M2 including AT, RNue, LUI, $\{IMSI\}_{inv(PK_{ap})}$, and HNID. The M2 is encrypted by SK_{apws}.

3. WAAA → LAAA (M3): The WAAA decrypts M2 with SKapws. The WAAA server forwards the message M3 to LAAA server. The information in M3 are AT, LUI, RNue, {IMSI}_inv (PKls), and HNID. The M3 is encrypted SKwsls.

4. LAAA → HSS (M4): The LAAA server forwards the M4 to HSS in LTE network. The information in M4 are AT, RNue, LUI, and {IMSI}_inv (PKls). The M4 is encrypted by PKhss.

5. HSS → LAAA (M5): The HSS checks RNls for freshness and find the information of the user according to IMSI value. Then, the HSS makes LUI and compares it with received LUI. It also generates Last Network ID (LNI) and New Key (NK): $LNI = LNIGFunc(RNhss, LUI, SK)$, $NK = NKFunc(SK, (LUI \oplus LNI))$. It uses NK to calculate MAC, XRES, CK, IK, AK, KASME, and AUTN: $MAC = F1(SQN || RAND || AMF)NK$, $XRES = F2(RAND)NK$, $CK = F3(RAND)NK$, $IK = F4(RAND)NK$, $AK = F5(RAND)NK$, $KASME = KDF((SQN \oplus AK), HNID, CK, IK, NK)$, $AUTN = AUTHFunc((SQN \oplus AK) || AMF || MAC)$. The HSS sets M5 including RNhss, RAND, AUTN, KASME, LNI, Pkue, and {XRES}_inv (PKhss). The M5 is encrypted by PKls.

6. LAAA → WAAA (M6): The LAAA saves received AV (i.e., RAND, AUTN, and KASME) and sends RNhss, RAND, AUTN, KASME, LNI, XRES, and PKue using M6 encrypted by SKswls.

7. WAAA → AP (M7): The WAAA server saves KASME and forwards RNhss, RAND, AUTN, LNI, and PKue via M7 encrypted by SKapws.

8. AP → UE (M8): The AP sets M8 including RAND, AUTN, {LNI}_inv(PK_{ap}), and RN_{hss}. It encrypts these parameters by PK_{ue} and sends M8 to UE.

9. UE → AP (M9): The UE makes LNI and NK: $LNI = LNIGFunc(RN_{hss}, LUI, SK)$, $NK = NKFunc(SK, (LUI \oplus LNI))$. It uses NK to generate CK, MAC, RES, AK, IK, AUTN, and KASME same as HSS. Then, the UE sets M9 including {RES}_inv(PK_{ue}). The M9 is encrypted by PK_{ap}.

10. AP → WAAA (M10): The AP forwards {RES}_inv(PK_{ue}) to WAAA server in M10 encrypted by SK_{apws}.

11. WAAA → AP (M11): The WAAA server compares the received RES with saved XRES to authenticate UE. Then, it makes MK: $MK = KDF(KASME, RN_{ws})$ and sends M11 including RN_{ws}, AKA success, MK. This is encrypted by SK_{apws}.

12. AP → UE (M12): The AP saves MK and forwards “AKA success” and RN_{ws} value to the UE via M12 encrypted by PK_{ue}.

13. UE: The UE checks the “AKA success” in M12 and makes MK same as WAAA.

6.2.5 REPS-AKA4 Ph2.2.2

This protocol provides authentication when the UE has been connected to LWI system before and it tries to connect to the system again. In fact, this is its n^{th} attachment. The system uses the security information that is saved in LAAA server. The steps of the protocol are presented as follows (Figure 54):

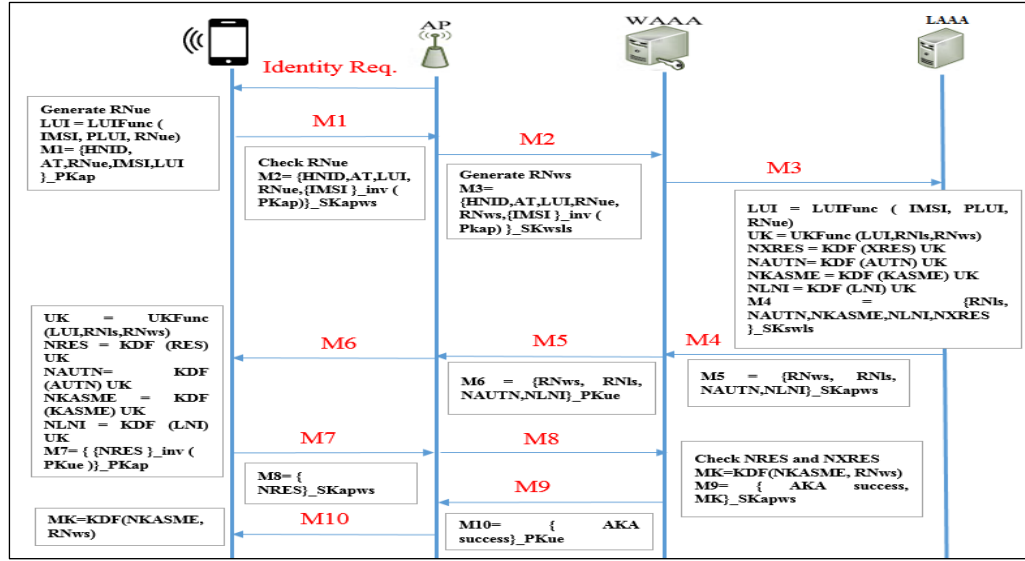


Figure 54: Steps of REPS-AKA4 Ph2.2.2

1. UE → AP (M1): When a user (i.e., UE) tries to connect to LWI network, and it has previous identifier in LWI network, the UE identifies itself by transmitting its identifier via M1 message. The UE generates a RNue and uses IMSI, Pervious LUI (PLUI), and RNue to make LUI: $LUI = LUIFunc(IMSI, PLUI, RNue)$ and sets AT field to 3. The LTE-WLAN network use this filed to figure out that it must use REPS-AKA4 Ph2.2.2 protocol for AKA procedure. The UE set HNID to inform the system that it is belong to which LTE network. Consequently, the UE transmits M1 including HNID, AT, RNue, IMSI, and LUI. The M1 message is encrypted by PKap.

2. AP → WAAA (M2): The AP decrypts M1 and checks RNue to prove freshness of the received message. The AP forwards M2 including HNID, AT, LUI, RNue, and $\{IMSI\}_{inv}(PKap)$. The message M2 is encrypted by SKapws.

3. WAAA → LAAA (M3): The WAAA decrypts M2 with its secret key SKapws and generates RNws. The WAAA server forwards the message M3 to LAAA server. The information in M3 are AT, RNws, LUI, RNue, $\{IMSI\}_{inv}(PKls)$, and HNID.

4. LAAA → WAAA (M4): The LAAA uses saved AV (i.e., RAND, AUTN, KASME, and XRES) and LNI from first attach. The LAAA makes LUI and compares it with received one. It generates Updated Key (UK) using LUI and generated RNls: $UK = UKFunc(LUI, RNls, RNws)$. It also uses UK to make NXRES, NAUTN, NKASME, and NLNI: $NXRES = KDF(XRES) UK$, $NAUTN = KDF(AUTN) UK$, $NKASME = KDF(KASME) UK$, $NLNI = KDF(LNI) UK$. Then, the LAAA sets M4 including RNls, NAUTN, NKASME, NLNI, and NXRES. The M4 message is encrypted by SKswls.

5. WAAA → AP (M5): The WAAA server sends M5 including RNws, RNls, NAUTN, and NLNI. This message is encrypted by SKapws.

6. AP → UE (M6): The AP sends M6 including RNws, RNls, NAUTN, and NLNI encrypted by PKue.

7. UE → AP (M7): The UE generates UK using LUI: $UK = UKFunc(LUI, RNls, RNws)$. Then, it uses UK to make NRES, NAUTN, NKASME, and NLNI same as LAAA. The UE sends M7 (which has {NRES}_inv (PKue)) encrypted by PKap to AP.

8. AP → WAAA (M8): The AP forwards NRES to WAAA server via M8 encrypted by SKapws.

9. WAAA → AP (M9): The WAAA server compares the received NRES with saved NXRES to confirm UE and makes MK: $MK = KDF(KASME, RNws)$. Then, it sends M9 including AKA success, and MK. This message is encrypted by SKapws.

10. AP → UE (M10): The AP saves MK and forwards “AKA success” to UE via M10 encrypted by PKue.

11. UE: The UE checks the “AKA success” in received M12 and makes MK same as WAAA.

6.2.6 REPS-AKA4 Ph3.1

The intra-handover is used when users (i.e., UEs) moves between different APs which are in the same WLAN area. In fact both target and source AP are connected to a same WAAA. The system employs REPS-AKA4 Ph3.1 when the UE attached to LTE network through eNodeB (by REPS-AKA4 Ph1.1 or REPS-AKA4 Ph1.2), then it connected to AP1 (by REPS-AKA4 Ph2.1) and then it moves to AP2 (i.e., target AP). The steps of the REPS-AKA4 Ph3.1 protocol are presented as follows (Figure 55):

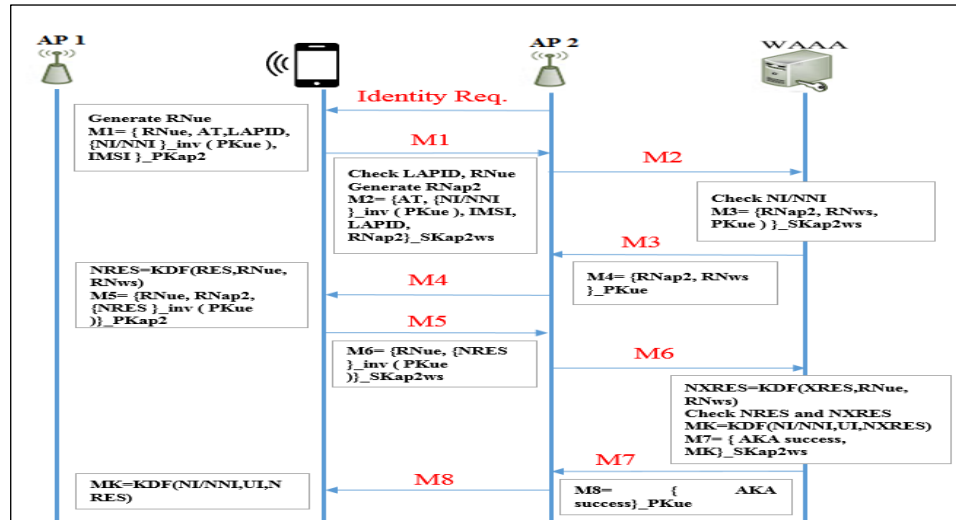


Figure 55: Steps of REPS-AKA4 Ph3.1

1. UE → AP2 (M1): When an UE tries to connect to new AP (i.e., AP2) in same WLAN and it has previous identifier in LWI network, the UE identifies itself by transmitting a valid parameter (i.e., NI/NNI) via M1 message. The UE generates an

RNue. Then, it sets AT field to 4. The LTE-WLAN network use this field to figure out that it must use REPS-AKA4 Ph3.1 protocol for AKA procedure. The UE set Last Access Point ID (LAPID) to inform network that it belongs to which AP. Consequently, the UE transmits M1 including RNue, AT, LAPID, {NI/NNI}_inv (PKue), and IMSI. The M1 is encrypted by PKap2 which is public key of AP2.

2. AP2 → WAAA (M2): The AP2 decrypts message M1 and checks the received RNue and LAPID. Then, it generates Random Number (RNap2). The AP2 forwards M2 including AT, {NI/NNI}_inv (PKue), IMSI, LAPID, and RNap2. The M2 message is encrypted by pre-shared key (SKap2ws).

3. WAAA → AP2 (M3): The WAAA server checks NI/NNI and sends M3 including RNap2, RNws, and PKue. This message is encrypted by SKap2ws.

4. AP2 → UE (M4): The AP2 sends message M4 including RNap2, and RNws. This message is encrypted by PKue.

5. UE → AP2 (M5): The UE generates NRES using RES, RNue, and RNws: $NRES = KDF(RES, RNue, RNws)$. Then, the UE sets message M5 which is encrypted by PKap2. The M5 includes RNue, RNap2, and {NRES}_inv (PKue).

6. AP2 → WAAA (M6): The AP2 forwards RNue and {NRES}_inv (PKue) to WAAA server via M6 encrypted by SKap2ws.

7. WAAA → AP2 (M7): The WAAA server makes NXRES: $NXRES = KDF(XRES, RNue, RNws)$. It compares the received NRES with generated NXRES to confirm UE

and makes MK: $MK = KDF(NI/NNI, UI, NXRES)$. Then, it sends M7 including AKA success, and MK. This message is encrypted by SKap2ws.

8. AP2 → UE (M8): The AP2 saves MK and forwards “AKA success” to UE via M8 encrypted by PKue.

9. UE: The UE checks the “AKA success” in received M8 and makes MK: $MK = KDF(NI/NNI, UI, NRES)$.

6.2.7 REPS-AKA4 Ph3.2

When the UE first attaches to LWI through LTE network (i.e. by REPS-AKA4 Ph2.2.1 or REPS-AKA4 Ph2.2.2) and then moves from AP1 to AP2 in same WLAN, the system employs REPS-AKA4 Ph3.2 protocol. The steps of the protocol are presented follows (Figure 56):

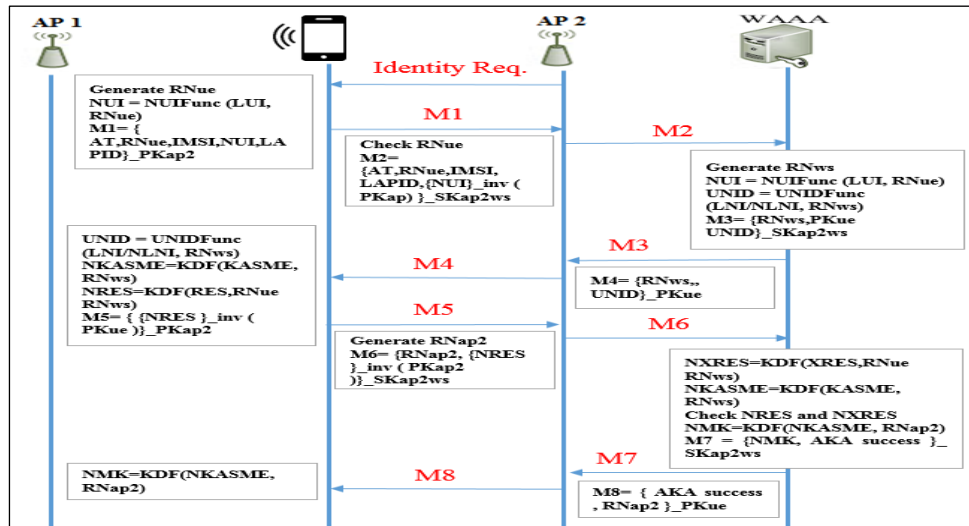


Figure 56: Steps of REPS-AKA4 Ph3.2

1. UE → AP2 (M1): When the UE tries to connect to new AP2 in same WLAN and it has previous identifier in LWI network, the UE transmits its valid identifier via M1

message. The UE generates an RNue. Then, it sets AT field to 5. The LTE-WLAN network use this field to learn that it must use REPS-AKA4 Ph3.2 protocol for AKA procedure. The UE generates New User Identity (NUI) using Last User Identity (LUI) and RNue: $NUI = NUIFunc(LUI, RNue)$. Then, the UE transmits M1 message including AT, RNue, IMSI, NUI, and LAPID. The M1 is encrypted by PKap2.

2. AP2 → WAAA (M2): The AP2 decrypts M1 and checks the received RNue to prove freshness of the M1. Then, the AP2 forwards M2 including AT, RNue, IMSI, LAPID, and $\{NUI\}_{inv}(PKap)$. The message M2 is encrypted by SKap2ws.

3. WAAA → AP2 (M3): The WAAA server makes NUI and compares generated NUI with received one. The WAAA generates RNws and calculates Updated Network ID (UNID): $UNID = UNIDFunc(LNI/NLNI, RNws)$. Then, it sends M3 including RNws, PKue and UNID encrypted by SKap2ws.

4. AP2 → UE (M4): The AP2 sends M4 including UNID, and RNws. This message is encrypted by PKue.

5. UE → AP2 (M5): The UE makes UNID and compares it with received one. It also generates NKASME and NRES using RES, KASME, RNue, and RNws: $NKASME = KDF(KASME, RNws)$, $NRES = KDF(RES, RNue, RNws)$. Then, sends M5 including $\{NRES\}_{inv}(PKue)$ encrypted by PKap2.

6. AP2 → WAAA (M6): The AP2 generates RNap2 and forwards RNap2 and $\{NRES\}_{inv}(PKap2)$ to WAAA server in M6 encrypted by SKap2ws.

7. WAAA → AP2 (M7): The WAAA server makes NXRES: $NXRES = KDF(XRES, RNue, RNws)$. It compares the received NRES with generated NXRES to confirm UE and makes NKASME and New MK (NMK): $NKASME = KDF(KASME, RNws)$, $NMK = KDF(NKASME, RNaP2)$. Then, it sends M7 including NMK, and AKA success. This message is encrypted by SKap2ws.

8. AP2 → UE (M8): The AP2 saves MK and forwards “AKA success”, and RNaP2 to UE via M8 encrypted by PKue.

9. UE: The UE checks the “AKA success” in M8 and makes NMK same as WAAA.

6.2.8 REPS-AKA4 Ph4.1

The inter-handover occurs when the UE moves to different APs which are in different WLANs. In this case, the UE is attached to LTE network by REPS-AKA4 Ph1.1 or REPS-AKA4 Ph1.2, and then it is connected to WLAN (i.e., AP1) by REPS-AKA4 Ph2.1, then it changes its AP to new AP (i.e., AP3) which is placed in another WLAN. The details of this protocol are presented in the following steps (Figure 57):

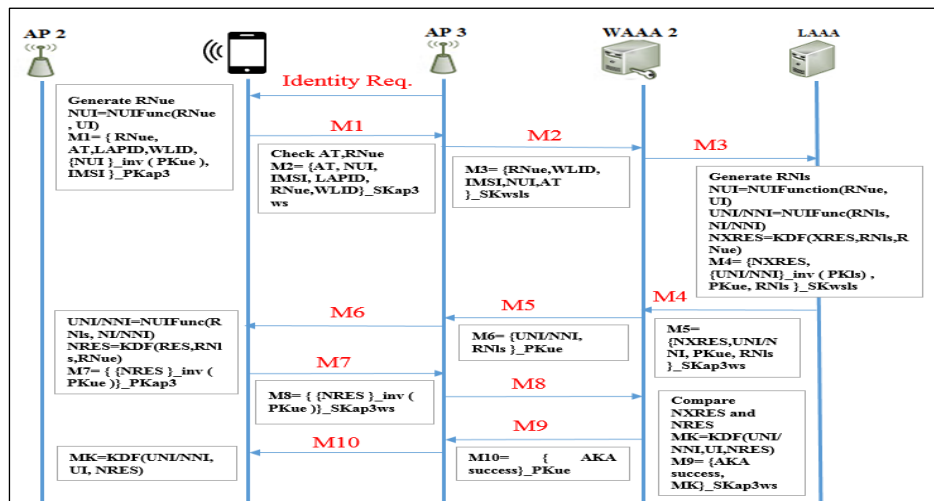


Figure 57: Steps of REPS-AKA4 Ph4.1

1. UE → AP3 (M1): The UE generates a RNue and calculates New UI (NUI): $NUI = NUIFunction(RNue, UI)$ and sets AT field to 6. The network uses this field to learn that it must use REPS-AKA4 Ph4.1 protocol for AKA procedure. The UE sets LAPID and WLAN ID (WLID) to inform the network that the UE was in area of which AP and WLAN. Consequently, the UE transmits M1 including RNue, AT, LAPID, WLID, $\{NUI\}_{inv}(PKue)$, and IMSI. The message M1 is encrypted by PKap3 which is public key of AP3.

2. AP3 → WAAA2 (M2): The AP3 decrypts message M1 and checks the received RNue and AT. The AP3 forwards M2 including AT, NUI, IMSI, LAPID, RNue, and WLID. The message M2 is encrypted by pre-shared key (SKap3ws).

3. WAAA2 → LAAA (M3): The WAAA2 server forwards the M3 to LAAA server. The information in M3 are RNue, WLID, IMSI, NUI, and AT encrypted by SKwsls.

4. LAAA → WAAA2 (M4): The LAAA uses saved XRES, NI/NNI and UI from previous attaches. The LAAA makes NUI and compares it with received one. It generates Updated NI/NNI (UNI/NNI) using generated Random Number (RNls): $UNI/NNI = NUIFunction(RNls, NI/NNI)$. It also uses XRES to make NXRES: $NXRES = KDF(XRES, RNls, RNue)$. Then, the LAAA sets M4 including NXRES, $\{UNI/NNI\}_{inv}(PKls)$, PKue, and RNls. The M4 message is encrypted by SKswls.

5. WAAA2 → AP3 (M5): The WAAA server sends M5 including NXRES, UNI/NNI, PKue, and RNls. This message is encrypted by SKap3ws.

6. AP2 → UE (M6): The AP2 sends M6 including UNI/NNI, and RNIs. This message is encrypted by PKue.

7. UE → AP2 (M7): The UE generates UNI/NNI using NI/NNI: $UNI/NNI = NUIFunction(RNIs, NI/NNI)$. It also uses RES to make NRES: $NRES = KDF(RES, RNIs, RNue)$. Then, the UE sets M7 including $\{NRES\}_{inv}$ (PKue). The message M7 is encrypted by PKap3.

8. AP3 → WAAA2 (M8): The AP3 forwards $\{NRES\}_{inv}$ (PKue) to WAAA2 server via M8 encrypted by SKap3ws.

9. WAAA2 → AP3 (M9): The WAAA2 server compares the value of the received NRES with saved NXRES to authenticate UE and makes MK: $MK = KDF(UNI/NNI, UI, NRES)$. Then, it sends M9 including “AKA success”, and MK. This message is encrypted by SKap3ws.

10. AP3 → UE (M10): The AP3 saves MK and forwards “AKA success” to UE using M10 encrypted by PKue.

11. UE: The UE checks the “AKA success” in M10 and makes MK same as WAAA2.

6.2.9 REPS-AKA4 Ph4.2

When the UE is connected to LTE-WLAN by REPS-AKA4 Ph2.2.1 or REPS-AKA4 Ph2.2.2, and then it moves to next access point (i.e., AP3) that belongs to another WLAN, the system employs REPS-AKA4 Ph4.2. The steps of the REPS-AKA4 Ph4.2 protocol are presented in the following steps (Figure 58):

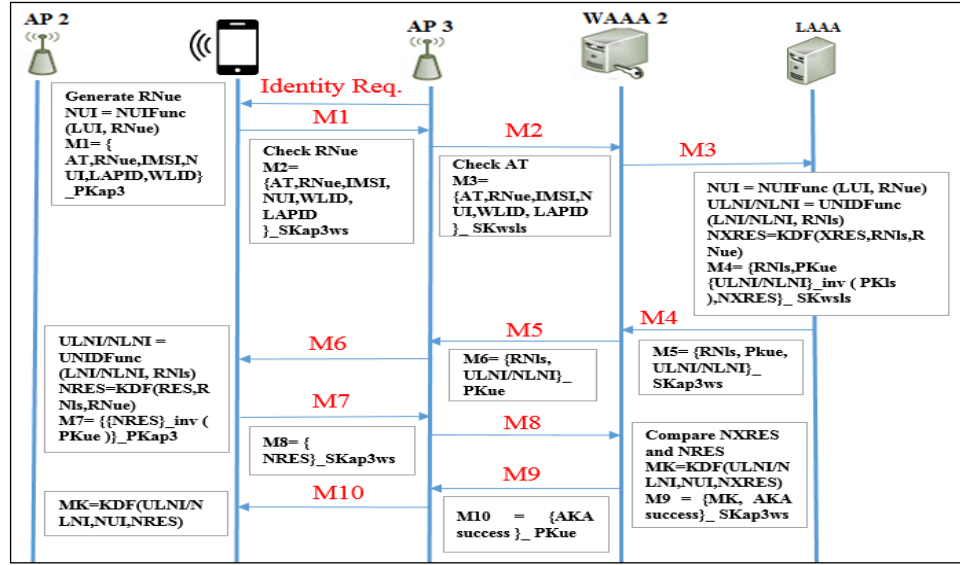


Figure 58: Steps of REPS-AKA4 Ph4.2

1. UE → AP3 (M1): The UE generates its RNue and makes NUI: $NUI = NUIFunction(RNue, UI)$ and sets AT field. The network uses this filed to learn that it must use REPS-AKA4 Ph4.2 protocol for AKA procedure. The UE sets LAPID and WLID and transmits M1 including RNue, AT, LAPID, WLID, NUI, and IMSI. The message M1 is encrypted by PKap3.

2. AP3 → WAAA2 (M2): The AP3 decrypts message M1 by its private key and checks the received RNue and AT. Then, it forwards M2 including AT, RNue, IMSI, NUI, WLID, and LAPID. The M2 is encrypted by pre-shared key (SKap3ws).

3. WAAA2 → LAAA (M3): The WAAA2 forwards the M3 (encrypted by SKswls) to LAAA. The information in M3 are AT, RNue, IMSI, NUI, WLID, and LAPID.

4. LAAA → WAAA2 (M4): The LAAA makes NUI and compares it with received one. Then, the LAAA uses saved XRES, NI/NNI from pervious attaches to make ULNI/NLNI and NXRES: $ULNI/NLNI = UNIDFunc(LNI/NLNI, RNIs)$, $NXRES =$

KDF (XRES, RNls, RNue). The LAAA sets M4 including RNls, PKue, {ULNI/NLNI}_inv(PKls), and NXRES. The message M4 is encrypted by SKswls.

5. WAAA2 → AP3 (M5): The WAAA2 server sends M5 including RNls, PKue, and ULNI/NLNI. This message is encrypted by SKap3ws.

6. AP2 → UE (M6): The AP3 sends M6 including RNls, RNap2, and ULNI/NLNI. This message is encrypted by PKue.

7. UE → AP3 (M7): The UE generates ULNI/NLNI using LNI/NLNI: $ULNI/NLNI = UNIDFunc(LNI/NLNI, RNls)$. It also uses RES to calculate NRES: $NRES = KDF(RES, RNls, RNue)$. Then, the UE sets M7 including NRES. The message M7 is encrypted by PKap2.

8. AP3 → WAAA2 (M8): The AP3 forwards NRES to WAAA2 server in M8 encrypted by SKap3ws.

9. WAAA2 → AP3 (M9): The WAAA2 server compares the value of received NRES with saved NXRES to authenticate UE and generates MK: $MK = KDF(ULNI/NLNI, NUI, NXRES)$. Then, it sends M9 including AKA success, and MK. This message is encrypted by SKap3ws.

10. AP3 → UE (M10): The AP3 saves MK and forwards “AKA success” to UE via M10 encrypted by PKue.

11. UE: The UE checks the “AKA success” in M10 and generates MK.

6.3 Security analysis of REPS-AKA4 protocol

The REPS-AKA4 protocols are evaluated according to security parameters in this section. The Security analyses of our methods are conducted using BAN Logic, AVISPA, Scyther tools.

6.3.1 Formal analysis using AVISPA

We have implemented the proposed AKA protocols by AVISPA tool. Figure 59 and 60 indicate the evaluation results of the REPS-AKA4 protocols in OFMC and CL-AtSe modes respectively (Details are in Appendix1). Here, ‘‘SAFE’’ is the keyword showing that the simulated protocols are safe according to the defined security goals. (AVISPA codes are presented in Appendix 1).

% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph1.1.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.03s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph1.2.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.01s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.1.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.02s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.2.1.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.06s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.2.2.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.02s visitedNodes: 2 nodes depth: 1 piles
% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph3.1.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.02s visitedNodes: 3 nodes depth: 2 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph3.2.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.01s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph4.1.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.02s visitedNodes: 2 nodes depth: 1 piles	% OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph4.2.f GOAL as specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.02s visitedNodes: 2 nodes depth: 1 piles	

Figure 59: Evaluation results of REPS AKA4 in OFMC mode

SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph1.1.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 5 states Reachable : 1 states Translation: 0.01 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph1.2.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 3 states Reachable : 1 states Translation: 0.00 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.1.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 5 states Reachable : 0 states Translation: 0.05 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.2.1.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 5 states Reachable : 0 states Translation: 0.10 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph2.2.2.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 3 states Reachable : 0 states Translation: 0.04 seconds Computation: 0.00 seconds
SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph3.1.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 5 states Reachable : 1 states Translation: 0.02 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph3.2.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 1 states Reachable : 0 states Translation: 0.01 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph4.1.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 9 states Reachable : 1 states Translation: 0.03 seconds Computation: 0.00 seconds	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/testsuite/results/REPS-AKA4.ph4.2.f GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 4 states Reachable : 0 states Translation: 0.02 seconds Computation: 0.00 seconds	

Figure 60: Evaluation results of REPS-AKA4 in CL-AtSe mode

6.3.2 Formal analysis using Scyther

In this section, we evaluate the proposed REPS-AKA4 protocols by means of Scyther tool. In the modeled protocols, five claims (i.e., Nisynch, Secrecy, Weakagree, Niagree, and Alive) are checked for mentioned nodes. (Scyther codes are presented in Appendix 2).

Based on UE's viewpoint in REPS-AKA4 Ph1.1, the security claims are presented as follows: *claim_ue1 (UE, Secret, IMSI); claim_ue2 (UE, Secret, RNue); claim_ue3 (UE, Secret, UI); claim_ue4 (UE, Secret, TS); claim_ue5 (UE, Secret, RES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);*

Based on MME's viewpoint in REPS-AKA4 Ph1.1, the security claims are presented as follows: *claim_mme1 (MME, Secret, RNmme); claim_mme2 (MME, Secret, RNue); claim_mme3 (MME, Secret, HNID); claim_mme4 (MME, Secret, IMSI); claim_mme5 (MME, Secret, UI); claim_mme6 (MME, Secret, RAND); claim_mme7 (MME, Secret, AUTN); claim_mme8 (MME, Secret, RNhss); claim_mme9 (MME, Secret, NI); claim_mmea (MME, Alive); claim_mmew (MME, Weakagree); claim_mmenia (MME, Niagree); claim_mmenis (MME, Nisynch);*

Based on HSS's viewpoint in REPS-AKA4 Ph1.1, the security claims are presented as follows: *claim_hss1 (HSS, Secret, NI); claim_hss2 (HSS, Secret, RAND); claim_hss3 (HSS, Secret, AUTN); claim_hss4 (HSS, Secret, KASME); claim_hss5 (HSS, Secret, XRES); claim_hss6 (HSS, Secret, RNhss); claim_hss7 (HSS, Secret, RNmme); claim_hssa (HSS, Alive); claim_hssw (HSS, Weakagree); claim_hssnia (HSS, Niagree); claim_hssnis (HSS, Nisynch);*

Based on UE's viewpoint in REPS-AKA4 Ph1.2, the security claims are presented as follows: *claim_ue1 (UE, Secret, IMSI); claim_ue2 (UE, Secret, RNue); claim_ue3 (UE, Secret, NUI); claim_ue4 (UE, Secret, NRES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);*

Based on MME's viewpoint in REPS-AKA4 Ph1.2, the security claims are presented as follows: *claim_mme1* (MME, Secret, RNue); *claim_mme2* (MME, Secret, NUI); *claim_mme3* (MME, Secret, IMSI); *claim_mme4* (MME, Secret, RNmme); *claim_mme5* (MME, Secret, UAUTN); *claim_mme6* (MME, Secret, URAND); *claim_mmea* (MME, Alive); *claim_mmew* (MME, Weakagree); *claim_mmenia* (MME, Niagree); *claim_mmenis* (MME, Nisynch);

Based on HSS's viewpoint in REPS-AKA4 Ph1.2, the security claims are presented as follows: *claim_hssa* (HSS, Alive); *claim_hssw* (HSS, Weakagree); *claim_hssnia* (HSS, Niagree); *claim_hssnis* (HSS, Nisynch);

Based on UE's viewpoint in REPS-AKA4 Ph2.1, the security claims are presented as follows: *claim_ue1* (UE, Secret, RNue); *claim_ue2* (UE, Secret, HNID); *claim_ue3* (UE, Secret, AT); *claim_ue4* (UE, Secret, NINNI); *claim_ue5* (UE, Secret, IMSI); *claim_ue6* (UE, Secret, RNap); *claim_ue7* (UE, Secret, NRES); *claim_uea* (UE, Alive); *claim_uew* (UE, Weakagree); *claim_uenia* (UE, Niagree); *claim_uenis* (UE, Nisynch);

Based on AP's viewpoint in REPS-AKA4 Ph2.1, the security claims are presented as follows: *claim_ap1* (AP, Secret, HNID); *claim_ap2* (AP, Secret, AT); *claim_ap3* (AP, Secret, NINNI); *claim_ap4* (AP, Secret, RNls); *claim_ap5* (AP, Secret, IMSI); *claim_ap6* (AP, Secret, APID); *claim_ap7* (AP, Secret, UI); *claim_ap8* (AP, Secret, RNap); *claim_ap9* (AP, Secret, NRES); *claim_ap10* (AP, Secret, AKASuccess); *claim_apa* (AP, Alive); *claim_apw* (AP, Weakagree); *claim_apnia* (AP, Niagree); *claim_apnis* (AP, Nisynch);

Based on WAAA's viewpoint in REPS-AKA4 Ph2.1, the security claims are presented as follows: *claim_waaa1* (WAAA, Secret, AT); *claim_waaa2* (WAAA, Secret, NINNI); *claim_waaa3* (WAAA, Secret, UI); *claim_waaa4* (WAAA, Secret, IMSI); *claim_waaa5* (WAAA, Secret, HNID); *claim_waaa6* (WAAA, Secret, RNls); *claim_waaa7* (WAAA, Secret, MK); *claim_waaa8* (WAAA, Secret,

AKASuccess);*claim_waaaa* (WAAA, Alive); *claim_waaaw* (WAAA, Weakagree); *claim_waaania* (WAAA, Niagree);*claim_waaanis* (WAAA, Nisynch);

Based on LAAA's viewpoint in REPS-AKA4 Ph2.1, the security claims are presented as follows: *claim_laaa1* (LAAA, Secret, RNIs); *claim_laaa2* (LAAA, Secret, UI); *claim_laaa3* (LAAA, Secret, XRES); *claim_laaa4* (LAAA, Secret, IMSI);*claim_laaaa* (LAAA, Alive); *claim_laaaw* (LAAA, Weakagree); *claim_laaania* (LAAA, Niagree);*claim_laaanis* (LAAA, Nisynch);

Based on HSS's viewpoint in REPS-AKA4 Ph2.1, the security claims are presented as follows: *claim_hss1* (HSS, Secret, RNIs); *claim_hss2* (HSS, Secret, XRES); *claim_hss3* (HSS, Secret, NINNI); *claim_hss4* (HSS, Secret, UI); *claim_hssa* (HSS, Alive); *claim_hssw* (HSS, Weakagree); *claim_hssnia* (HSS, Niagree);*claim_hssnis* (HSS, Nisynch);

Based on UE's viewpoint in REPS-AKA4 Ph2.2.1, the security claims are presented as follows: *claim_ue1* (UE, Secret, HNID); *claim_ue2* (UE, Secret, TS); *claim_ue3* (UE, Secret, AT); *claim_ue4* (UE, Secret, RNue); *claim_ue5* (UE, Secret, IMSI); *claim_ue6* (UE, Secret, LUI);*claim_ue7* (UE, Secret, RES); *claim_uea* (UE, Alive); *claim_uew* (UE, Weakagree); *claim_uenia* (UE, Niagree); *claim_uenis* (UE, Nisynch);

Based on AP's viewpoint in REPS-AKA4 Ph2.2.1, the security claims are presented as follows: *claim_ap1* (AP, Secret, HNID); *claim_ap2* (AP, Secret, AT); *claim_ap3* (AP, Secret, LUI); *claim_ap4* (AP, Secret, RNue); *claim_ap5* (AP, Secret, IMSI); *claim_ap6* (AP, Secret, RES); *claim_ap7* (AP, Secret, RAND); *claim_ap8* (AP, Secret, AUTN); *claim_ap9* (AP, Secret, LNI); *claim_ap10* (AP, Secret, RNhss); *claim_ap11* (AP, Secret, AKASuccess); *claim_ap12* (AP, Secret, RNws); *claim_apa* (AP, Alive); *claim_apw* (AP, Weakagree); *claim_apnia* (AP, Niagree);*claim_apnis* (AP, Nisynch);

Based on WAAA's viewpoint in REPS-AKA4 Ph2.2.1, the security claims are presented as follows: *claim_waaa1* (WAAA, Secret, AT); *claim_waaa2* (WAAA, Secret, LUI);

claim_waaa3 (WAAA, Secret, RNue); claim_waaa4 (WAAA, Secret, IMSI); claim_waaa5 (WAAA, Secret, HNID); claim_waaa6 (WAAA, Secret, RNhss); claim_waaa7 (WAAA, Secret, RAND); claim_waaa8 (WAAA, Secret, AUTN); claim_waaa9 (WAAA, Secret, LNI); claim_waaa10 (WAAA, Secret, RNhss); claim_waaa11 (WAAA, Secret, AKASuccess); claim_waaa12 (WAAA, Secret, MK); claim_waaaa (WAAA, Alive); claim_waaaw (WAAA, Weakagree); claim_waaania (WAAA, Niagree); claim_waaanis (WAAA, Nisynch);

Based on LAAA's viewpoint in REPS-AKA4 Ph2.2.1, the security claims are presented as follows: *claim_laaa1 (LAAA, Secret, AT); claim_laaa2 (LAAA, Secret, RNue); claim_laaa3 (LAAA, Secret, LUI); claim_laaa4 (LAAA, Secret, IMSI); claim_laaa5 (LAAA, Secret, RNhss); claim_laaa6 (LAAA, Secret, RAND); claim_laaa7 (LAAA, Secret, AUTN); claim_laaa8 (LAAA, Secret, KASME); claim_laaa9 (LAAA, Secret, LNI); claim_laaa10 (LAAA, Secret, RES); claim_laaaa (LAAA, Alive); claim_laaaw (LAAA, Weakagree); claim_laaania (LAAA, Niagree); claim_laaanis (LAAA, Nisynch);*

Based on HSS's viewpoint in REPS-AKA4 Ph2.2.1, the security claims are presented as follows: *claim_hss1 (HSS, Secret, RNhss); claim_hss2 (HSS, Secret, RAND); claim_hss3 (HSS, Secret, AUTN); claim_hss4 (HSS, Secret, KASME); claim_hss5 (HSS, Secret, LNI); claim_hss6 (HSS, Secret, RES); claim_hssa (HSS, Alive); claim_hssw (HSS, Weakagree); claim_hssnia (HSS, Niagree); claim_hssnis (HSS, Nisynch);*

Based on UE's viewpoint in REPS-AKA4 Ph2.2.2, the security claims are presented as follows: *claim_ue1 (UE, Secret, HNID); claim_ue2 (UE, Secret, AT); claim_ue3 (UE, Secret, RNue); claim_ue4 (UE, Secret, LUI); claim_ue5 (UE, Secret, IMSI); claim_ue6 (UE, Secret, NRES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);*

Based on AP's viewpoint in REPS-AKA4 Ph2.2.2, the security claims are presented as follows: *claim_ap1 (AP, Secret, HNID); claim_ap2 (AP, Secret, AT); claim_ap3 (AP, Secret, LUI);*

claim_ap4 (AP, Secret, RNue); claim_ap5 (AP, Secret, IMSI); claim_ap6 (AP, Secret, RNws); claim_ap7 (AP, Secret, RNls); claim_ap8 (AP, Secret, NAUTN); claim_ap9 (AP, Secret, NLNI); claim_ap10 (AP, Secret, NRES); claim_ap11 (AP, Secret, AKASuccess); claim_apa (AP, Alive); claim_apw (AP, Weakagree); claim_apnia (AP, Niagree); claim_apnis (AP, Nisynch);

Based on WAAA's viewpoint in REPS-AKA4 Ph2.2.2, the security claims are presented as follows: *claim_waaa1 (WAAA, Secret, HNID); claim_waaa2 (WAAA, Secret, AT); claim_waaa3 (WAAA, Secret, LUI); claim_waaa4 (WAAA, Secret, RNue); claim_waaa5 (WAAA, Secret, RNws); claim_waaa6 (WAAA, Secret, IMSI); claim_waaa7 (WAAA, Secret, RNls); claim_waaa8 (WAAA, Secret, NAUTN); claim_waaa9 (WAAA, Secret, NLNI); claim_waaaa (WAAA, Alive); claim_waaaw (WAAA, Weakagree); claim_waaania (WAAA, Niagree); claim_waaanis (WAAA, Nisynch);*

Based on LAAA's viewpoint in REPS-AKA4 Ph2.2.2, the security claims are presented as follows: *claim_laaa1 (LAAA, Secret, RNls); claim_laaa2 (LAAA, Secret, NAUTN); claim_laaa3 (LAAA, Secret, NKASME); claim_laaa4 (LAAA, Secret, NLNI); claim_laaa5 (LAAA, Secret, NXRES); claim_laaaa (LAAA, Alive); claim_laaaw (LAAA, Weakagree); claim_laaania (LAAA, Niagree); claim_laaanis (LAAA, Nisynch);*

Based on UE's viewpoint in REPS-AKA4 Ph3.1, the security claims are presented as follows: *claim_ue1 (UE, Secret, RNue); claim_ue2 (UE, Secret, AT); claim_ue3 (UE, Secret, LAPID); claim_ue4 (UE, Secret, NINNI); claim_ue5 (UE, Secret, IMSI); claim_ue6 (UE, Secret, RNap2); claim_ue7 (UE, Secret, NRES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);*

Based on AP2's viewpoint in REPS-AKA4 Ph3.1, the security claims are presented as follows: *claim_ap21 (AP2, Secret, RNue); claim_ap22 (AP2, Secret, AT); claim_ap23 (AP2, Secret, LAPID); claim_ap24 (AP2, Secret, NINNI); claim_ap25 (AP2, Secret, IMSI); claim_ap26 (AP2, Secret, RNap2); claim_ap27 (AP2, Secret, RNws); claim_ap28 (AP2, Secret, NRES); claim_ap29 (AP2, Secret,*

AKASuccess); claim_ap2a (AP2, Alive); claim_ap2w (AP2, Weakagree); claim_ap2nia (AP2, Niagree);claim_ap2nis (AP2, Nisynch);

Based on WAAA's viewpoint in REPS-AKA4 Ph3.1, the security claims are presented as follows: *claim_waaa1 (WAAA, Secret, RNap2); claim_waaa2 (WAAA, Secret, RNws); claim_waaa3 (WAAA, Secret, AKASuccess); claim_waaa4 (WAAA, Secret, MK); claim_waaaa (WAAA, Alive); claim_waaaw (WAAA, Weakagree); claim_waaania (WAAA, Niagree);claim_waaanis (WAAA, Nisynch);*

Based on UE's viewpoint in REPS-AKA4 Ph3.2, the security claims are presented as follows: *claim_ue1 (UE, Secret, RNue); claim_ue2 (UE, Secret, AT); claim_ue3 (UE, Secret, LAPID); claim_ue4 (UE, Secret, NUI); claim_ue5 (UE, Secret, IMSI); claim_ue6 (UE, Secret, NRES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);*

Based on AP2's viewpoint in REPS-AKA4 Ph3.2, the security claims are presented as follows: *claim_ap21 (AP2, Secret, RNue); claim_ap22 (AP2, Secret, AT); claim_ap23 (AP2, Secret, LAPID); claim_ap24 (AP2, Secret, NUI); claim_ap25 (AP2, Secret, IMSI); claim_ap26 (AP2, Secret, RNap2); claim_ap27 (AP2, Secret, RNws); claim_ap28 (AP2, Secret, NRES);claim_ap29 (AP2, Secret, AKASuccess);claim_ap210 (AP2, Secret, UNID); claim_ap2a (AP2, Alive); claim_ap2w (AP2, Weakagree); claim_ap2nia (AP2, Niagree);claim_ap2nis (AP2, Nisynch);*

Based on WAAA's viewpoint in REPS-AKA4 Ph3.2, the security claims are presented as follows: *claim_waaa1 (WAAA, Secret, RNws); claim_waaa2 (WAAA, Secret, UNID); claim_waaa3 (WAAA, Secret, AKASuccess); claim_waaa4 (WAAA, Secret, NMK);claim_waaaa (WAAA, Alive); claim_waaaw (WAAA, Weakagree); claim_waaania (WAAA, Niagree);claim_waaanis (WAAA, Nisynch);*

Based on UE's viewpoint in REPS-AKA4 Ph4.1, the security claims are presented as follows: *claim_ue1 (UE, Secret, RNue)*; *claim_ue2 (UE, Secret, AT)*; *claim_ue3 (UE, Secret, LAPID)*; *claim_ue4 (UE, Secret, WLID)*; *claim_ue5 (UE, Secret, IMSI)*; *claim_ue6 (UE, Secret, NUI)*; *claim_ue7 (UE, Secret, NRES)*; *claim_uea (UE, Alive)*; *claim_uew (UE, Weakagree)*; *claim_uenia (UE, Niagree)*; *claim_uenis (UE, Nisynch)*;

Based on AP3's viewpoint in REPS-AKA4 Ph4.1, the security claims are presented as follows: *claim_ap31 (AP3, Secret, RNue)*; *claim_ap32 (AP3, Secret, AT)*; *claim_ap33 (AP3, Secret, LAPID)*; *claim_ap34 (AP3, Secret, UNINNI)*; *claim_ap35 (AP3, Secret, IMSI)*; *claim_ap36 (AP3, Secret, NUI)*; *claim_ap37 (AP3, Secret, WLID)*; *claim_ap38 (AP3, Secret, NRES)*; *claim_ap39 (AP3, Secret, AKASuccess)*; *claim_ap310 (AP3, Secret, RNIs)*; *claim_ap3a (AP3, Alive)*; *claim_ap3w (AP3, Weakagree)*; *claim_ap3nia (AP3, Niagree)*; *claim_ap3nis (AP3, Nisynch)*;

Based on WAAA2's viewpoint in REPS-AKA4 Ph4.1, the security claims are presented as follows: *claim_waaa21 (WAAA2, Secret, RNue)*; *claim_waaa22 (WAAA2, Secret, WLID)*; *claim_waaa23 (WAAA2, Secret, IMSI)*; *claim_waaa24 (WAAA2, Secret, NUI)*; *claim_waaa25 (WAAA2, Secret, AT)*; *claim_waaa26 (WAAA2, Secret, UNINNI)*; *claim_waaa27 (WAAA2, Secret, RNIs)*; *claim_waaa28 (WAAA2, Secret, AKASuccess)*; *claim_waaaa2 (WAAA2, Alive)*; *claim_waaa2w (WAAA2, Weakagree)*; *claim_waaa2nia (WAAA2, Niagree)*; *claim_waaa2nis (WAAA2, Nisynch)*;

Based on LAAA's viewpoint in REPS-AKA4 Ph4.1, the security claims are presented as follows: *claim_laaa1 (LAAA, Secret, NXRES)*; *claim_laaa2 (LAAA, Secret, UNINNI)*; *claim_laaa3 (LAAA, Secret, RNIs)*; *claim_laaaa (LAAA, Alive)*; *claim_laaaw (LAAA, Weakagree)*; *claim_laaania (LAAA, Niagree)*; *claim_laaanis (LAAA, Nisynch)*;

Based on UE's viewpoint in REPS-AKA4 Ph4.2, the security claims are presented as follows: *claim_ue1 (UE, Secret, RNue)*; *claim_ue2 (UE, Secret, AT)*; *claim_ue3 (UE, Secret, LAPID)*; *claim_ue4 (UE, Secret, WLID)*; *claim_ue5 (UE, Secret, IMSI)*; *claim_ue6 (UE, Secret, NUI)*; *claim_ue7*

(UE, Secret, NRES); claim_uea (UE, Alive); claim_uew (UE, Weakagree); claim_uenia (UE, Niagree); claim_uenis (UE, Nisynch);

Based on AP3's viewpoint in REPS-AKA4 Ph4.2, the security claims are presented as follows:*claim_ap31 (AP3, Secret, RNue); claim_ap32 (AP3, Secret, AT); claim_ap33 (AP3, Secret, LAPID); claim_ap34 (AP3, Secret, ULNINLNI); claim_ap35 (AP3, Secret, IMSI); claim_ap36 (AP3, Secret, NUI); claim_ap37 (AP3, Secret, WLID); claim_ap38 (AP3, Secret, NRES); claim_ap39 (AP3, Secret, AKASuccess); claim_ap310 (AP3, Secret, RNIs); claim_ap3a (AP3, Alive); claim_ap3w (AP3, Weakagree); claim_ap3nia (AP3, Niagree); claim_ap3nis (AP3, Nisynch);*

Based on WAAA2's viewpoint in REPS-AKA4 Ph4.2, the security claims are presented as follows: *claim_waaa21 (WAAA2, Secret, RNue); claim_waaa22 (WAAA2, Secret, WLID); claim_waaa23 (WAAA2, Secret, IMSI); claim_waaa24 (WAAA2, Secret, NUI); claim_waaa25 (WAAA2, Secret, AT); claim_waaa26 (WAAA2, Secret, ULNINLNI); claim_waaa27 (WAAA2, Secret, RNIs); claim_waaa28 (WAAA2, Secret, AKASuccess); claim_waaa29 (WAAA2, Secret, LAPID); claim_waaa210 (WAAA2, Secret, MK); claim_waaaa2 (WAAA2, Alive); claim_waaa2w (WAAA2, Weakagree); claim_waaa2nia (WAAA2, Niagree); claim_waaa2nis (WAAA2, Nisynch);*

Based on LAAA's viewpoint in REPS-AKA4 Ph4.2, the security claims are presented as follows:*claim_laaa1 (LAAA, Secret, NXRES); claim_laaa2 (LAAA, Secret, ULNINLNI); claim_laaa3 (LAAA, Secret, RNIs); claim_laaaa (LAAA, Alive); claim_laaaw (LAAA, Weakagree); claim_laaania (LAAA, Niagree); claim_laaanis (LAAA, Nisynch);*

The evaluation results of REPS-AKA4 Ph1.1 protocol are presented in Figures 61-63. As it can be seen in these figures, the proposed protocol satisfy all defined security climes in involved nodes. The evaluation results for other phases of REPS-AKA4 are the same as REPS-AKA4 Ph1.1.

Claim				Status	Comments
REPSAKA411	UE	REPSAKA411,ue1	Secret IMSI	Ok	Verified No attacks.
		REPSAKA411,ue2	Secret RNue	Ok	Verified No attacks.
		REPSAKA411,ue3	Secret UIFunc(IMSI,XOR(SK,RNue))	Ok	Verified No attacks.
		REPSAKA411,ue4	Secret TS	Ok	Verified No attacks.
		REPSAKA411,ue5	Secret F2(RAND,NKFunc(XOR(RNmme,NIGFunc(RNhss,UIG...	Ok	Verified No attacks.
		REPSAKA411,uea	Alive	Ok	Verified No attacks.
		REPSAKA411,uew	Weakagree	Ok	Verified No attacks.
		REPSAKA411,uenia	Niagree	Ok	Verified No attacks.
		REPSAKA411,uenis	Nisynch	Ok	Verified No attacks.

Figure 61: Scyther evaluation for UE (REPS-AKA4 Ph1.1)

Claim				Status	Comments
REPSAKA411	MME	REPSAKA411,mme1	Secret RNmme	Ok	Verified No attacks.
		REPSAKA411,mme2	Secret RNue	Ok	Verified No attacks.
		REPSAKA411,mme3	Secret NIFunc(XOR(MVP,HVP))	Ok	Verified No attacks.
		REPSAKA411,mme4	Secret IMSI	Ok	Verified No attacks.
		REPSAKA411,mme5	Secret UIFunc(IMSI,XOR(SK,RNue))	Ok	Verified No attacks.
		REPSAKA411,mme6	Secret RAND	Ok	Verified No attacks.
		REPSAKA411,mme7	Secret AUTHFunc(XOR(SQN,F5(RAND,NKFunc(XOR(RNmme,...	Ok	Verified No attacks.
		REPSAKA411,mme8	Secret RNhss	Ok	Verified No attacks.
		REPSAKA411,mme9	Secret NIGFunc(RNhss,UIFunc(IMSI,XOR(SK,RNue)),SK...	Ok	Verified No attacks.
		REPSAKA411,mmea	Alive	Ok	Verified No attacks.
		REPSAKA411,mmev	Weakagree	Ok	Verified No attacks.
		REPSAKA411,mmenia	Niagree	Ok	Verified No attacks.
		REPSAKA411,mmenis	Nisynch	Ok	Verified No attacks.

Figure 62: Scyther evaluation for MME (REPS-AKA4 Ph1.1)

Claim				Status	Comments
REPSAKA411	HSS	REPSAKA411,hss1	Secret NIGFunc(RNhss,UIGFunc(IMSI,XOR(SK,RNue)),SK...	Ok	Verified No attacks.
		REPSAKA411,hss2	Secret RAND	Ok	Verified No attacks.
		REPSAKA411,hss3	Secret AUTHFunc(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,...	Ok	Verified No attacks.
		REPSAKA411,hss4	Secret KDF(XOR(SQN,F5(RAND,NKGFunc(XOR(RNmme,NIGFu...	Ok	Verified No attacks.
		REPSAKA411,hss5	Secret F2(RAND,NKGFunc(XOR(RNmme,NIGFunc(RNhss,UIG...	Ok	Verified No attacks.
		REPSAKA411,hss6	Secret RNhss	Ok	Verified No attacks.
		REPSAKA411,hss7	Secret RNmme	Ok	Verified No attacks.
		REPSAKA411,hssa	Alive	Ok	Verified No attacks.
		REPSAKA411,hssw	Weakagree	Ok	Verified No attacks.
		REPSAKA411,hssnia	Niagree	Ok	Verified No attacks.
		REPSAKA411,hssnis	Nisynch	Ok	Verified No attacks.

Figure 63: Scyther evaluation for HSS (REPS-AKA4 Ph1.1)

6.3.3 Security analysis by BAN logic

We prove the AKA procedure of the presented REPS-AKA4 methods using the BAN Logic. This logic properly verifies the transmitted messages and checks confidentiality among involved nodes in modeled AKA methods. The formal messages of the suggested REPS-AKA4 protocols are as follows:

REPS-AKA4 Ph1.1: $M1: UE \rightarrow MME: \{\{IMSI\}_{inv(PKue)}, RNue, UI, PKue, TS\}_{PKmme}$, $M2: MME \rightarrow HSS: \{RNmme, RNue, \{HNID\}_{inv(PKmme)}, IMSI, UI\}_{SKmh}$, $M3: HSS \rightarrow MME: \{\{NI\}_{inv(PKhss)}, RAND, AUTN, KASME, XRES, RNhss, RNmme\}_{SKmh}$, $M4: MME \rightarrow UE: \{RAND, AUTN, RNhss, RNmme, HNID, \{NI\}_{inv(PKmme)}\}_{PKue}$, $M5: UE \rightarrow MME: \{RNmme, \{RES\}_{inv(PKue)}\}_{PKmme}$.

REPS-AKA4 Ph1.2: $Re-M1: UE \rightarrow MME: \{\{NUI\}_{inv(PKue)}, RNue, IMSI\}_{PKmme}$, $Re-M2: MME \rightarrow HSS: \{RNue, \{NUI\}_{inv(PKmme)}, IMSI\}_{SKmh}$, $Re-M3: MME \rightarrow UE: \{RNmme, RNue, \{UAUTN\}_{inv(PKmme)}, URAND\}_{PKue}$, $Re-M4: UE \rightarrow MME: \{RNmme, \{NRES\}_{inv(PKue)}\}_{PKmme}$.

REPS-AKA4 Ph2.1: **M1:** $UE \rightarrow AP: \{RNue, HNID, AT, \{NI/NNI\}_{inv}(PKue), IMSI}\}_{PKap}$, **M2:** $AP \rightarrow WAAA: \{AT, HNID, \{NI/NNI\}_{inv}(PKue), IMSI, APID}\}_{SKapws}$, **M3:** $WAAA \rightarrow LAAA: \{AT, HNID, \{NI/NNI\}_{inv}(PKue), IMSI}\}_{SKwsls}$, **M4:** $LAAA \rightarrow HSS: \{RNls, \{IMSI\}_{inv}(PKls)}\}_{PKhss}$, **M5:** $HSS \rightarrow LAAA: \{RNls, XRES, PKue, \{NI/NNI\}_{inv}(PKhss), UI}\}_{PKls}$, **M6:** $LAAA \rightarrow WAAA: \{NXRES, RNls, PKue, UI}\}_{SKwsls}$, **M7:** $WAAA \rightarrow AP: \{RNls, PKue, UI}\}_{SKapws}$, **M8:** $AP \rightarrow UE: \{RNls, UI, RNap}\}_{PKue}$, **M9:** $UE \rightarrow AP: \{RNap, \{NRES\}_{inv}(PKue)}\}_{PKap}$, **M10:** $AP \rightarrow WAAA: \{\{NRES\}_{inv}(PKue)}\}_{SKapws}$, **M11:** $WAAA \rightarrow AP: \{AKASuccess, MK}\}_{SKapws}$, **M12:** $AP \rightarrow WAAA: \{AKASuccess}\}_{PKue}$.

REPS-AKA4 Ph2.2.1: **M1:** $UE \rightarrow AP: \{HNID, TS, AT, RNue, IMSI, LUI}\}_{PKap}$, **M2:** $AP \rightarrow WAAA: \{HNID, AT, LUI, RNue, \{IMSI\}_{inv}(PKap)}\}_{SKapws}$, **M3:** $WAAA \rightarrow LAAA: \{AT, LUI, RNue, IMSI, HNID}\}_{SKwsls}$, **M4:** $LAAA \rightarrow HSS: \{AT, RNue, LUI, \{IMSI\}_{inv}(PKls)}\}_{PKhss}$, **M5:** $HSS \rightarrow LAAA: \{RNhss, RAND, AUTN, KASME, LNI, PKue, \{XRES\}_{inv}(PKhss)}\}_{PKls}$, **M6:** $LAAA \rightarrow WAAA: \{RNhss, RAND, AUTN, KASME, LNI, XRES, PKue}\}_{SKwsls}$, **M7:** $WAAA \rightarrow AP: \{RNhss, RAND, AUTN, LNI, PKue}\}_{SKapws}$, **M8:** $AP \rightarrow UE: \{RAND, AUTN, \{LNI\}_{inv}(PKap), RNhss}\}_{PKue}$, **M9:** $UE \rightarrow AP: \{\{RES\}_{inv}(PKue)}\}_{PKap}$, **M10:** $AP \rightarrow WAAA: \{RES\}_{SKapws}$, **M11:** $WAAA \rightarrow AP: \{RNws, AKASuccess, MK}\}_{SKapws}$, **M12:** $AP \rightarrow UE: \{AKASuccess, RNws}\}_{PKue}$.

REPS-AKA4 Ph2.2.2: **M1:** $UE \rightarrow AP: \{HNID, TS, RNue, IMSI, LUI}\}_{PKap}$, **M2:** $AP \rightarrow WAAA: \{HNID, AT, LUI, RNue, \{IMSI\}_{inv}(PKap)}\}_{SKapws}$, **M3:** $WAAA \rightarrow LAAA: \{HNID, AT, LUI, RNue, RNws, \{IMSI\}_{inv}(PKap)}\}_{SKwsls}$, **M4:** $LAAA \rightarrow WAAA: \{RNls, NAUTN, NKASME, NLNI, NXRES}\}_{SKwsls}$, **M5:** $WAAA \rightarrow AP: \{RNws, RNls, NAUTN, NLNI}\}_{SKapws}$, **M6:** $AP \rightarrow UE: \{RNws, RNls, NAUTN, NLNI}\}_{PKue}$, **M7:** $UE \rightarrow AP: \{\{NRES\}_{inv}(PKue)}\}_{PKap}$, **M8:** $AP \rightarrow WAAA: \{NRES\}_{SKapws}$, **M9:** $WAAA \rightarrow AP: \{AKASuccess, MK}\}_{SKapws}$, **M10:** $AP \rightarrow UE: \{AKASuccess}\}_{PKue}$.

REPS-AKA4 Ph3.1: **M1:** $UE \rightarrow AP2: \{RNue, AT, LAPID, \{NI/NNI\}_{inv}(PKue), IMSI}\}_{PKap2}$, **M2:** $AP2 \rightarrow WAAA: \{AT, \{NI/NNI\}_{inv}(PKue), IMSI, LAPID, RNap2}\}_{SKap2ws}$, **M3:** $WAAA \rightarrow AP2: \{RNap2, RNws, PKue}\}_{SKap2ws}$, **M4:** $AP2 \rightarrow UE: \{RNap2, RNws}\}_{PKue}$, **M5:** $UE \rightarrow AP2:$

$\{RNue, RNap2, \{NRES\}_{inv(PKue)}\}_{PKap2}, M6: AP2 \rightarrow WAAA: \{RNue, \{NRES\}_{inv(PKue)}\}_{SKap2ws},$
 $M7: WAAA \rightarrow AP2: \{AKASuccess, MK\}_{SKap2ws}, M8: AP2 \rightarrow UE: \{AKASuccess\}_{PKue}.$

REPS-AKA4 Ph3.2: $M1: UE \rightarrow AP2: \{AT, RNue, IMSI, NUI, LAPID\}_{PKap2}, M2: AP2 \rightarrow$
 $WAAA: \{AT, RNue, IMSI, LAPID, \{NUI\}_{inv(PKap2)}\}_{SKap2ws}, M3: WAAA \rightarrow AP2: \{RNws, PKue,$
 $UNID\}_{SKap2ws}, M4: AP2 \rightarrow UE: \{RNws, UNID\}_{PKue}, M5: UE \rightarrow AP2:$
 $\{\{NRES\}_{inv(PKue)}\}_{PKap2}, M6: AP2 \rightarrow WAAA: \{RNap2, \{NRES\}_{inv(PKap2)}\}_{SKap2ws}, M7:$
 $WAAA \rightarrow AP2: \{NMK, AKASuccess\}_{SKap2ws}, M8: AP2 \rightarrow UE: \{AKASuccess, RNap2\}_{PKue}.$

REPS-AKA4 Ph4.1: $M1: UE \rightarrow AP3: \{RNue, AT, LAPID, WLID, \{NUI\}_{inv(PKue)},$
 $IMSI\}_{PKap3}, M2: AP3 \rightarrow WAAA2: \{AT, NUI, IMSI, LAPID, RNue, WLID\}_{SKap3ws}, M3: WAAA2$
 $\rightarrow LAAA: \{RNue, WLID, IMSI, NUI, AT\}_{SKws}, M4: LAAA \rightarrow WAAA2: \{NXRES,$
 $\{UNI/NNI\}_{inv(PKls)}, PKue, RNls\}_{SKws}, M5: WAAA2 \rightarrow AP3: \{NXRES, UNI/NNI, PKue,$
 $RNls\}_{SKap3ws}, M6: AP3 \rightarrow UE: \{UNI/NNI, RNls\}_{PKue}, M7: UE \rightarrow AP3:$
 $\{\{NRES\}_{inv(PKue)}\}_{PKap3}, M8: AP3 \rightarrow WAAA2: \{\{NRES\}_{inv(PKue)}\}_{SKap3ws}, M9: WAAA2 \rightarrow$
 $AP3: \{AKASuccess, MK\}_{SKap3ws}, M10: AP3 \rightarrow UE: \{AKASuccess\}_{PKue}.$

REPS-AKA4 Ph4.2: $M1: UE \rightarrow AP3: \{AT, RNue, IMSI, NUI, LAPID, WLID\}_{PKap3}, M2: AP3$
 $\rightarrow WAAA2: \{AT, RNue, IMSI, NUI, WLID, LAPID\}_{SKap3ws}, M3: WAAA2 \rightarrow LAAA: \{AT, RNue,$
 $IMSI, NUI, WLID, LAPID\}_{SKws}, M4: LAAA \rightarrow WAAA2: \{RNls, PKue, \{ULNI/NLNI\}_{inv(PKls)},$
 $NXRES\}_{SKws}, M5: WAAA2 \rightarrow AP3: \{RNls, PKue, ULNI/NLNI\}_{SKap3ws}, M6: AP3 \rightarrow UE: \{RNls,$
 $ULNI/NLNI\}_{PKue}, M7: UE \rightarrow AP3: \{\{NRES\}_{inv(PKue)}\}_{PKap3}, M8: AP3 \rightarrow WAAA2:$
 $\{\{NRES\}_{inv(PKue)}\}_{SKap3ws}, M9: WAAA2 \rightarrow AP3: \{MK, AKASuccess\}_{SKap3ws}, M10: AP3 \rightarrow$
 $UE: \{AKASuccess\}_{PKue}.$

We need to transform all messages into idealized logical format as follows:

REPS-AKA4 Ph1.1: $M1: UE \rightarrow MME: MME \triangleleft \{\{IMSI\}_{PKue^{-1}}, UE \mid \Rightarrow \#(RNue), UE \mid \Rightarrow UI,$
 $PKue, \#(TS)\}_{PKmme}, M2: MME \rightarrow HSS: HSS \triangleleft \{MME \mid \Rightarrow \#(RNmme), UE \mid \Rightarrow \#(RNue),$
 $\{MME \mid \Rightarrow HNID\}_{PKmme^{-1}}, IMSI, UE \mid \Rightarrow UI\}_{SKmh}, M3: HSS \rightarrow MME: MME \triangleleft \{\{HSS \mid \Rightarrow NI\}_{PKhss^{-1}},$

$HSS \Rightarrow \#(RAND)$, $HSS \Rightarrow AUTN$, $HSS \Rightarrow KASME$, $HSS \Rightarrow XRES$, $HSS \Rightarrow \#(RNhss)$,
 $MME \Rightarrow \#(RNmme)\}SKmh$, **M4: MME $\rightarrow$ UE:** $UE \triangleleft \{HSS \Rightarrow \#(RAND), HSS \Rightarrow AUTN,$
 $HSS \Rightarrow \#(RNhss), MME \Rightarrow \#(RNmme), MME \Rightarrow HNID, \{HSS \Rightarrow NI\}PKmme^{-1})\}PKue$, **M5: UE $\rightarrow$**
MME: $MME \triangleleft \{MME \Rightarrow \#(RNmme), \{UE \Rightarrow RES\}PKue^{-1}\}PKmme$.

REPS-AKA4 Ph1.2: Re-M1: UE $\rightarrow$ MME: $MME \triangleleft \{\{UE \Rightarrow NUI\}PKue^{-1}, UE \Rightarrow \#(RNue),$
 $IMSI\}PKmme$, **Re-M2: MME $\rightarrow$ HSS:** $HSS \triangleleft \{UE \Rightarrow \#(RNue), \{MME \Rightarrow NUI\}PKmme^{-1}, IMSI\}SKmh$,
Re-M3: MME $\rightarrow$ UE: $UE \triangleleft \{MME \Rightarrow \#(RNmme), UE \Rightarrow \#(RNue), \{MME \Rightarrow UAUTN\}PKmme^{-1},$
 $MME \Rightarrow URAND\}PKue$, **Re-M4: UE $\rightarrow$ MME:** $MME \triangleleft \{MME \Rightarrow \#(RNmme), \{UE \Rightarrow NRES\}PKue^{-1}\}PKmme$.

REPS-AKA4 Ph2.1: M1: UE $\rightarrow$ AP: $AP \triangleleft \{UE \Rightarrow \#(RNue), MME \Rightarrow HNID, UE \Rightarrow AT,$
 $\{UE \Rightarrow NI/NNI\}PKue^{-1}, IMSI\}PKap$, **M2: AP $\rightarrow$ WAAA:** $WAAA \triangleleft \{UE \Rightarrow AT, MME \Rightarrow HNID,$
 $\{UE \Rightarrow NI/NNI\}PKue^{-1}, IMSI, APID\}SKapws$, **M3: WAAA $\rightarrow$ LAAA:** $LAAA \triangleleft \{UE \Rightarrow AT,$
 $MME \Rightarrow HNID, \{UE \Rightarrow NI/NNI\}PKue^{-1}, IMSI\}SKwsls$, **M4: LAAA $\rightarrow$ HSS:** $HSS \triangleleft \{LAAA \Rightarrow \#(RNls),$
 $\{IMSI\}PKls^{-1}\}PKhss$, **M5: HSS $\rightarrow$ LAAA:** $LAAA \triangleleft \{LAAA \Rightarrow \#(RNls), HSS \Rightarrow XRES, PKue,$
 $\{HSS \Rightarrow NI/NNI\}PKhss^{-1}, UE \Rightarrow UI\}PKls$, **M6: LAAA $\rightarrow$ WAAA:** $WAAA \triangleleft \{LAAA \Rightarrow NXRES,$
 $LAAA \Rightarrow \#(RNls), PKue, UE \Rightarrow UI\}SKwsls$, **M7: WAAA $\rightarrow$ AP:** $AP \triangleleft \{LAAA \Rightarrow \#(RNls), PKue,$
 $UE \Rightarrow UI\}SKapws$, **M8: AP $\rightarrow$ UE:** $UE \triangleleft \{LAAA \Rightarrow \#(RNls), UE \Rightarrow UI, AP \Rightarrow \#(RNap)\}PKue$, **M9:**
UE $\rightarrow$ AP: $AP \triangleleft \{LAAA \Rightarrow \#(RNap), \{UE \Rightarrow NRES\}PKue^{-1}\}PKap$, **M10: AP $\rightarrow$ WAAA:** $WAAA \triangleleft$
 $\{\{UE \Rightarrow NRES\}PKue^{-1}\}SKapws$, **M11: WAAA $\rightarrow$ AP:** $AP \triangleleft \{AKASuccess, WAAA \Rightarrow MK\}SKapws$,
M12: AP $\rightarrow$ UE: $\{AKASuccess\}PKue$.

REPS-AKA4 Ph2.2.1: M1: UE $\rightarrow$ AP: $AP \triangleleft \{MME \Rightarrow HNID, UE \Rightarrow TS, UE \Rightarrow AT,$
 $UE \Rightarrow \#(RNue), IMSI, UE \Rightarrow LUI\}PKap$, **M2: AP $\rightarrow$ WAAA:** $WAAA \triangleleft \{MME \Rightarrow HNID, UE \Rightarrow AT,$
 $UE \Rightarrow LUI, UE \Rightarrow \#(RNue), \{IMSI\}PKap^{-1}\}SKapws$, **M3: WAAA $\rightarrow$ LAAA:** $LAAA \triangleleft \{UE \Rightarrow AT,$
 $UE \Rightarrow LUI, UE \Rightarrow \#(RNue), IMSI, MME \Rightarrow HNID\}SKwsls$, **M4: LAAA $\rightarrow$ HSS:** $HSS \triangleleft \{UE \Rightarrow AT,$
 $UE \Rightarrow \#(RNue), UE \Rightarrow LUI, \{IMSI\}PKls^{-1}\}PKhss$, **M5: HSS $\rightarrow$ LAAA:** $LAAA \triangleleft \{HSS \Rightarrow \#(RNhss),$
 $HSS \Rightarrow RAND, HSS \Rightarrow AUTN, HSS \Rightarrow KASME, HSS \Rightarrow LNI, PKue, \{HSS \Rightarrow XRES\}PKhss^{-1}\}PKls$,

M6: LAAA $\rightarrow$ WAAA: WAAA $\triangleleft \{HSS| \Rightarrow \#(RNhss), HSS| \Rightarrow RAND, HSS| \Rightarrow AUTN, HSS| \Rightarrow KASME, HSS| \Rightarrow LNI, HSS| \Rightarrow XRES, PKue\}SKwsls$, **M7: WAAA $\rightarrow$ AP:** AP $\triangleleft \{HSS| \Rightarrow \#(RNhss), HSS| \Rightarrow RAND, HSS| \Rightarrow AUTN, HSS| \Rightarrow LNI, PKue\}SKapws$, **M8: AP $\rightarrow$ UE:** UE $\triangleleft \{HSS| \Rightarrow RAND, HSS| \Rightarrow AUTN, \{HSS| \Rightarrow LNI\}PKap^{-1}, HSS| \Rightarrow \#(RNhss)\}PKue$, **M9: UE $\rightarrow$ AP:** AP $\triangleleft \{\{UE| \Rightarrow RES\}PKue^{-1}\}PKap$, **M10: AP $\rightarrow$ WAAA:** WAAA $\triangleleft \{UE| \Rightarrow RES\}SKapws$, **M11: WAAA $\rightarrow$ AP:** AP $\triangleleft \{WAAA| \Rightarrow \#(RNws), AKASuccess, WAAA| \Rightarrow MK\}SKapws$, **M12: AP $\rightarrow$ UE:** UE $\triangleleft \{AKASuccess, WAAA| \Rightarrow \#(RNws)\}PKue$.

REPS-AKA4 Ph2.2.2:M1: UE $\rightarrow$ AP: AP $\triangleleft \{MME| \Rightarrow HNID, UE| \Rightarrow TS, UE| \Rightarrow \#(RNue), IMSI, UE| \Rightarrow LUI\}PKap$, **M2: AP $\rightarrow$ WAAA:** WAAA $\triangleleft \{MME| \Rightarrow HNID, UE| \Rightarrow AT, UE| \Rightarrow LUI, UE| \Rightarrow \#(RNue), \{IMSI\}PKap^{-1}\}SKapws$, **M3: WAAA $\rightarrow$ LAAA:** LAAA $\triangleleft \{MME| \Rightarrow HNID, UE| \Rightarrow AT, UE| \Rightarrow LUI, UE| \Rightarrow \#(RNue), WAAA| \Rightarrow \#(RNws), \{IMSI\}PKap^{-1}\}SKwsls$, **M4: LAAA $\rightarrow$ WAAA:** WAAA $\triangleleft \{LAAA| \Rightarrow \#(RNls), LAAA| \Rightarrow NAUTN, LAAA| \Rightarrow NKASME, LAAA| \Rightarrow NLNI, LAAA| \Rightarrow NXRES\}SKwsls$, **M5: WAAA $\rightarrow$ AP:** AP $\triangleleft \{WAAA| \Rightarrow \#(RNws), LAAA| \Rightarrow \#(RNls), LAAA| \Rightarrow NAUTN, LAAA| \Rightarrow NLNI\}SKapws$, **M6: AP $\rightarrow$ UE:** UE $\triangleleft \{WAAA| \Rightarrow \#(RNws), LAAA| \Rightarrow \#(RNls), LAAA| \Rightarrow NAUTN, LAAA| \Rightarrow NLNI\}PKue$, **M7: UE $\rightarrow$ AP:** AP $\triangleleft \{\{UE| \Rightarrow NRES\}_{inv}(PKue)\}PKap$, **M8: AP $\rightarrow$ WAAA:** WAAA $\triangleleft \{UE| \Rightarrow NRES\}SKapws$, **M9: WAAA $\rightarrow$ AP:** AP $\triangleleft \{AKASuccess, WAAA| \Rightarrow MK\}SKapws$, **M10: AP $\rightarrow$ UE:** UE $\triangleleft \{AKASuccess\}PKue$.

REPS-AKA4 Ph3.1:M1: UE $\rightarrow$ AP2: AP2 $\triangleleft \{UE| \Rightarrow \#(RNue), UE| \Rightarrow AT, LAPID, \{UE| \Rightarrow NI/NNI\}PKue^{-1}, IMSI\}PKap2$, **M2: AP2 $\rightarrow$ WAAA:** WAAA $\triangleleft \{UE| \Rightarrow AT, \{UE| \Rightarrow NI/NNI\}PKue^{-1}, IMSI, LAPID, AP2| \Rightarrow \#(RNap2)\}SKap2ws$, **M3: WAAA $\rightarrow$ AP2:** AP2 $\triangleleft \{AP2| \Rightarrow \#(RNap2), WAAA| \Rightarrow \#(RNws), PKue\}SKap2ws$, **M4: AP2 $\rightarrow$ UE:** UE $\triangleleft \{AP2| \Rightarrow \#(RNap2), WAAA| \Rightarrow \#(RNws)\}PKue$, **M5: UE $\rightarrow$ AP2:** AP2 $\triangleleft \{UE| \Rightarrow \#(RNue), AP2| \Rightarrow \#(RNap2), \{UE| \Rightarrow NRES\}PKue^{-1}\}PKap2$, **M6: AP2 $\rightarrow$ WAAA:** WAAA $\triangleleft \{UE| \Rightarrow \#(RNue), \{UE| \Rightarrow NRES\}PKue^{-1}\}SKap2ws$, **M7: WAAA $\rightarrow$ AP2:** AP2 $\triangleleft \{AKASuccess, WAAA| \Rightarrow MK\}SKap2ws$, **M8: AP2 $\rightarrow$ UE:** UE $\triangleleft \{AKASuccess\}PKue$.

REPS-AKA4 Ph3.2:M1: UE $\rightarrow$ AP2: AP2 $\triangleleft \{UE| \Rightarrow AT, UE| \Rightarrow \#(RNue), IMSI, UE| \Rightarrow NUI, LAPID\}PKap2$, **M2: AP2 $\rightarrow$ WAAA:** WAAA $\triangleleft \{UE| \Rightarrow AT, UE| \Rightarrow \#(RNue), IMSI, LAPID,$

$\{UE \Rightarrow NUI\}PKap2^{-1}\}SKap2ws$, **M3: WAAA $\rightarrow$ AP2:** $AP2 \triangleleft \{WAAA \Rightarrow \#(RNws), PKue, WAAA \Rightarrow UNID\}SKap2ws$, **M4: AP2 $\rightarrow$ UE:** $UE \triangleleft \{WAAA \Rightarrow \#(RNws), WAAA \Rightarrow UNID\}PKue$, **M5: UE $\rightarrow$ AP2:** $AP2 \triangleleft \{\{UE \Rightarrow NRES\}PKue^{-1}\}PKap2$, **M6: AP2 $\rightarrow$ WAAA:** $WAAA \triangleleft \{AP2 \Rightarrow \#(RNa2), \{UE \Rightarrow NRES\}PKap2^{-1}\}SKap2ws$, **M7: WAAA $\rightarrow$ AP2:** $AP2 \triangleleft \{WAAA \Rightarrow NMK, AKASuccess\}SKap2ws$, **M8: AP2 $\rightarrow$ UE:** $UE \triangleleft \{AKASuccess, AP2 \Rightarrow \#(RNa2)\}PKue$.

REPS-AKA4 Ph4.1: **M1: UE $\rightarrow$ AP3:** $\{UE \Rightarrow \#(RNue), UE \Rightarrow AT, LAPID, WLID, \{UE \Rightarrow NUI\}PKue^{-1}, IMSI\}PKap3$, **M2: AP3 $\rightarrow$ WAAA2:** $\{UE \Rightarrow AT, UE \Rightarrow NUI, IMSI, LAPID, UE \Rightarrow \#(RNue), WLID\}SKap3ws$, **M3: WAAA2 $\rightarrow$ LAAA:** $\{UE \Rightarrow \#(RNue), WLID, IMSI, UE \Rightarrow NUI, UE \Rightarrow AT\}SKwsls$, **M4: LAAA $\rightarrow$ WAAA2:** $\{LAAA \Rightarrow NXRES, \{LAAA \Rightarrow UNI/NNI\}PKls^{-1}, PKue, LAAA \Rightarrow \#(RNls)\}SKwsls$, **M5: WAAA2 $\rightarrow$ AP3:** $\{LAAA \Rightarrow NXRES, LAAA \Rightarrow UNI/NNI, PKue, LAAA \Rightarrow \#(RNls)\}SKap3ws$, **M6: AP3 $\rightarrow$ UE:** $\{LAAA \Rightarrow UNI/NNI, LAAA \Rightarrow \#(RNls)\}PKue$, **M7: UE $\rightarrow$ AP3:** $\{\{UE \Rightarrow NRES\}PKue^{-1}\}PKap3$, **M8: AP3 $\rightarrow$ WAAA2:** $\{\{UE \Rightarrow NRES\}PKue^{-1}\}SKap3ws$, **M9: WAAA2 $\rightarrow$ AP3:** $\{AKASuccess, WAAA2 \Rightarrow MK\}SKap3ws$, **M10: AP3 $\rightarrow$ UE:** $\{AKASuccess\}PKue$.

REPS-AKA4 Ph4.2: **M1: UE $\rightarrow$ AP3:** $\{UE \Rightarrow AT, UE \Rightarrow \#(RNue), IMSI, UE \Rightarrow NUI, LAPID, WLID\}PKap3$, **M2: AP3 $\rightarrow$ WAAA2:** $\{UE \Rightarrow AT, UE \Rightarrow \#(RNue), IMSI, UE \Rightarrow NUI, WLID, LAPID\}SKap3ws$, **M3: WAAA2 $\rightarrow$ LAAA:** $\{UE \Rightarrow AT, UE \Rightarrow \#(RNue), IMSI, UE \Rightarrow NUI, WLID, LAPID\}SKwsls$, **M4: LAAA $\rightarrow$ WAAA2:** $\{LAAA \Rightarrow \#(RNls), PKue, \{LAAA \Rightarrow ULNI/NLNI\}PKls^{-1}, LAAA \Rightarrow NXRES\}SKwsls$, **M5: WAAA2 $\rightarrow$ AP3:** $\{LAAA \Rightarrow \#(RNls), PKue, LAAA \Rightarrow ULNI/NLNI\}SKap3ws$, **M6: AP3 $\rightarrow$ UE:** $\{LAAA \Rightarrow \#(RNls), LAAA \Rightarrow ULNI/NLNI\}PKue$, **M7: UE $\rightarrow$ AP3:** $\{\{UE \Rightarrow NRES\}PKue^{-1}\}PKap3$, **M8: AP3 $\rightarrow$ WAAA2:** $\{\{UE \Rightarrow NRES\}PKue^{-1}\}SKap3ws$, **M9: WAAA2 $\rightarrow$ AP3:** $\{WAAA2 \Rightarrow MK, AKASuccess\}SKap3ws$, **M10: AP3 $\rightarrow$ UE:** $\{AKASuccess\}PKue$.

The following initial Assumptions (A) are considered to evaluate the suggested method: **A1:** It is considered that the secret keys (i.e., SKmh, SKapws, SKwsls, SKap2ws and SKap3ws) and public keys (PKue, PKmme, PKhss, PKap, PKls, PKap2 and PKap3) are shared between involved entities as follows: $UE \models \xrightarrow{PKue} UE \wedge \xrightarrow{PKmme} MME$

$$\begin{aligned}
& \wedge \xrightarrow{PKap} AP \wedge \xrightarrow{PKap2} AP2 \wedge \xrightarrow{PKap3} AP3 \wedge UE \xleftrightarrow{SK} HSS, \mathbf{MME} \models \xrightarrow{PKue} UE \wedge \xrightarrow{PKmme} MME \wedge \xrightarrow{PKhss} HSS \wedge \\
& MME \xleftrightarrow{SKmh} HSS, \mathbf{HSS} \models \xrightarrow{PKmme} MME \wedge \xrightarrow{PKhss} HSS \wedge MME \xleftrightarrow{SKmh} HSS \wedge \xrightarrow{PKls} LAAA \wedge UE \xleftrightarrow{SK} HSS, \mathbf{AP} \models \\
& \xrightarrow{PKap} AP \wedge \xrightarrow{PKue} UE \wedge AP \xleftrightarrow{SKapws} WAAA, \mathbf{AP2} \models \xrightarrow{PKap2} AP2 \wedge \xrightarrow{PKue} UE \wedge AP2 \xleftrightarrow{SKap2ws} WAAA, \mathbf{AP3} \models \\
& \xrightarrow{PKap3} AP3 \wedge \xrightarrow{PKue} UE \wedge AP3 \xleftrightarrow{SKap3ws} WAAA2, \mathbf{WAAA} \models AP \xleftrightarrow{SKapws} WAAA \wedge WAAA \xleftrightarrow{SKwsls} LAAA \wedge \xrightarrow{PKue} UE \\
& \wedge \xrightarrow{PKap} AP, \mathbf{WAAA2} \models AP3 \xleftrightarrow{SKap3ws} WAAA2 \wedge WAAA \xleftrightarrow{SKwsls} LAAA \wedge \xrightarrow{PKue} UE, \mathbf{LAAA} \models \xrightarrow{PKhss} HSS \wedge \\
& \xrightarrow{PKls} LAAA \wedge WAAA \xleftrightarrow{SKwsls} LAAA \wedge \xrightarrow{PKue} UE, \mathbf{A2}: \text{It is assumed that the entities calculate (i.e.,}
\end{aligned}$$

control) some security parameters and generated keys as follows: $UE \models UI \wedge NI \wedge UKG \wedge MAC \wedge RES \wedge CK \wedge IK \wedge AK \wedge KASME \wedge AUTN \wedge NUI \wedge NNI \wedge UPK \wedge UKASME \wedge URES \wedge UAUTN \wedge URAND \wedge NRES \wedge MK \wedge LUI \wedge LNI \wedge NK \wedge UK \wedge NAUTN \wedge NKASME \wedge NLNI \wedge AT \wedge UNID \wedge UNI \wedge NNI \wedge ULNI \wedge NLNI, \mathbf{MME} \models HNID \wedge NUI \wedge UPK \wedge UAUTN \wedge URAND \wedge UKASME \wedge NXRES, \mathbf{HSS} \models HNID \wedge UI \wedge NI \wedge UKG \wedge MAC \wedge XRES \wedge CK \wedge IK \wedge AK \wedge KASME \wedge AUTN \wedge NUI \wedge NNI \wedge LUI \wedge LNI \wedge NK, \mathbf{WAAA} \models NUI \wedge UNID \wedge NXRES \wedge NKASME \wedge NMK \wedge MK, \mathbf{WAAA2} \models MK, \mathbf{LAAA} \models NXRES \wedge LUI \wedge UK \wedge NAUTN \wedge NKASME \wedge NLNI \wedge NUI \wedge UNI \wedge NNI \wedge ULNI \wedge NLNI, \mathbf{A3}$: We consider that the entities generate some random value to use in key derivation and authentication functions as follows: $UE \models \#(RNue), \mathbf{MME} \models \#(RNmme), \mathbf{HSS} \models \#(RNhss), \mathbf{AP} \models \#(RNap), \mathbf{LAAA} \models \#(RNls), \mathbf{WAAA} \models \#(RNws), \mathbf{AP2} \models \#(RNap2), \mathbf{AP3} \models \#(RNap3), \mathbf{A4}$: It is considered that the entities have some security value related to other involved nodes as follows:

$$\begin{aligned}
& UE \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge (UE \xleftrightarrow{SQN} HSS) \wedge (UE \xleftrightarrow{AT} LAAA) \wedge (UE \xleftrightarrow{LAPID} WAAA) \wedge (UE \\
& \xleftrightarrow{WLID} LAAA), \mathbf{MME} \models (MME \xleftrightarrow{MVP} HSS) \wedge (MME \xleftrightarrow{HVP} HSS), \mathbf{HSS} \models (UE \xleftrightarrow{IMSI} HSS) \wedge (UE \xleftrightarrow{AMF} HSS) \wedge \\
& (UE \xleftrightarrow{SQN} HSS) \wedge (MME \xleftrightarrow{MVP} HSS) \wedge (MME \xleftrightarrow{HVP} HSS), \mathbf{AP} \models (AP \xleftrightarrow{APID} WAAA), \mathbf{WAAA} \models (UE \xleftrightarrow{LAPID} \\
& WAAA), \mathbf{LAAA} \models (UE \xleftrightarrow{WLID} LAAA).
\end{aligned}$$

The main security Goals (**G**) of the suggested protocol are as follows: providing mutual authentication between involved entities (i.e., UE, MME, HSS, AP, WAAA, and LAAA), and establishing strong session keys between user side and network side. Detailed security goals are represented as follows: **G1**: The mutual authentication

between involved nodes is ensured in the suggested approach as $UE \models HSS \models MME \wedge MME \models HSS \models UE$ then $UE \models MME \wedge MME \models UE$ and also as $UE \models LAAA \models WAAA \wedge WAAA \models LAAA \models UE$ then $UE \models WAAA \wedge WAAA \models UE$, **G2**: Key acceptance achieved between different defined entities. There is a matched MK (NMK) between the UE and WAAA (AP) to present a strong key agreement. As $UE \models WAAA(AP) \models MK(NMK) \wedge WAAA(AP) \models UE \models MK(NMK)$ then $UE \xleftrightarrow{MK(NMK)} WAAA(AP)$, **G3**: Message authentication between corresponding entities is satisfied as $MME \models UE \sim IMSI$ (in M1 of REPS-AKA Ph1.1), $HSS \models MME \sim HNID$ (in M2 of REPS-AKA Ph1.1), $MME \models HSS \sim NI$ (in M3 of REPS-AKA Ph1.1), $UE \models MME \sim NI$ (in M4 of REPS-AKA Ph1.1), $MME \models UE \sim RES$ (in M5 of REPS-AKA Ph1.1), $MME \models UE \sim NUI$ (in M1 of REPS-AKA Ph1.2), $HSS \models MME \sim NUI$ (in M2 of REPS-AKA Ph1.2), $UE \models MME \sim UAUTN$ (in M3 of REPS-AKA Ph1.2), $MME \models UE \sim NRES$ (in M4 of REPS-AKA Ph1.2), $AP \models UE \sim NI/NNI$ (in M1 of REPS-AKA Ph2.1), $WAAA \models AP \sim NI/NNI$ (in M2 of REPS-AKA Ph2.1), $LAAA \models WAAA \sim NI/NNI$ (in M3 of REPS-AKA Ph2.1), $HSS \models LAAA \sim IMSI$ (in M4 of REPS-AKA Ph2.1), $LAAA \models HSS \sim NI/NNI$ (in M5 of REPS-AKA Ph2.1), $AP \models UE \sim NRES$ (in M9 of REPS-AKA Ph2.1), $WAAA \models AP \sim NRES$ (in M10 of REPS-AKA Ph2.1), $WAAA \models AP \sim IMSI$ (in M2 of REPS-AKA Ph2.2.1), $HSS \models LAAA \sim IMSI$ (in M4 of REPS-AKA Ph2.2.1), $LAAA \models HSS \sim XRES$ (in M5 of REPS-AKA Ph2.2.1), $UE \models AP \sim LNI$ (in M8 of REPS-AKA Ph2.2.1), $AP \models UE \sim RES$ (in M9 of REPS-AKA Ph2.2.1), $WAAA \models AP \sim IMSI$ (in M2 of REPS-AKA Ph2.2.2), $LAAA \models WAAA \sim IMSI$ (in M3 of REPS-AKA Ph2.2.2), $AP2 \models UE \sim NI/NNI$ (in M1 of REPS-AKA Ph3.1), $WAAA \models AP2 \sim NI/NNI$ (in M2 of REPS-AKA Ph3.1), $AP2 \models UE \sim NRES$ (in M5 of REPS-AKA Ph3.1), $WAAA \models AP2 \sim NRES$ (in M6 of REPS-AKA Ph3.1), $WAAA \models AP2 \sim NUI$ (in M2 of REPS-AKA Ph3.2), $AP2 \models UE \sim NRES$ (in M5 of REPS-AKA Ph3.2), $WAAA \models AP2 \sim NRES$ (in M6 of REPS-AKA Ph3.2), $AP3 \models UE \sim NUI$ (in M1 of REPS-AKA Ph4.1), $WAAA2 \models LAAA \sim$

UNI/NNI (in M4 of REPS-AKA Ph4.1), $AP3 \models UE \sim NRES$ (in M7 of REPS-AKA Ph4.1), $WAAA2 \models AP3 \sim NRES$ (in M8 of REPS-AKA Ph4.1), $WAAA2 \models LAAA \sim ULNI/NLNI$ (in M4 of REPS-AKA Ph4.2), $AP3 \models UE \sim NRES$ (in M7 of REPS-AKA Ph4.2), $WAAA2 \models AP3 \sim NRES$ (in M8 of REPS-AKA Ph4.2), **G4**: The key freshness between UE and WAAA is provided. The key freshness between these entities holds since $UE \models WAAA \models \#(RNws)$, $WAAA \models UE \models \#(RNws)$, $UE \models WAAA \models \#(RNaP2)$, $WAAA \models UE \models \#(RNaP2)$ thus, MK = KDF (NKASME, RNws), NMK = KDF (NKASME, RNaP2) are fresh keys accordingly, **G5**: Confidentiality between the UE (as a user side) and the AP (WAAA) (as a network side): $\frac{UE \models (UE \xrightarrow{MK(NMK)} AP), AP \triangleleft \{X\}_{MK(NMK)}}{UE \models AP \sim \{X\}_{MK(NMK)}} \wedge \frac{AP \models (UE \xrightarrow{MK(NMK)} AP), UE \triangleleft \{Y\}_{MK(NMK)}}{AP \models UE \sim \{Y\}_{MK(NMK)}}$ Where ‘X’ and ‘Y’ are transmitted messages from AP and UE respectively.

Then, we evaluate the idealized form of our presented method based on defined hypothesis and the BAN Logic rules. The used BAN Logic rules includes Message meaning rule (R1), Jurisdiction rule (R2), and Nonce Verification rule (R3).

We employ the BAN logic rule (R2) to derive:

$$\frac{HSS \models UE \Rightarrow RES, HSS \models UE \models RES}{HSS \models RES}, \frac{UE \models HSS \Rightarrow RES, UE \models HSS \models RES}{UE \models RES},$$

$$\frac{HSS \models UE \Rightarrow UI, HSS \models UE \models UI}{HSS \models UI}, \frac{UE \models HSS \Rightarrow NI, UE \models HSS \models NI}{UE \models NI} \text{ To satisfy } \mathbf{G1}. \text{ We utilize the BAN logic}$$

$$\text{rule } (\mathbf{R1}) \text{ to derive: } \frac{AP \models (AP \xrightarrow{SKapws} WAAA), AP \triangleleft \{MK(NMK)\}_{SKapws}}{AP \models WAAA \sim MK(NMK)} \wedge WAAA \models MK(NMK) \text{ To}$$

satisfy **G2**. For REPS-AKA4Ph1.1, we use the BAN logic rule (**R1**) to derive:

$$\frac{MME \models (PKue \xrightarrow{UE}) MME \triangleleft \{IMSI\}_{PKue^{-1}}}{MME \models UE \sim IMSI}, \frac{HSS \models (PKmme \xrightarrow{MME}) HSS \triangleleft \{HNID\}_{PKmme^{-1}}}{HSS \models MME \sim HNID},$$

$$\frac{MME \models (PKhss \xrightarrow{HSS}) MME \triangleleft \{NI\}_{PKhss^{-1}}}{MME \models HSS \sim NI}, \frac{UE \models (PKmme \xrightarrow{MME}) UE \triangleleft \{NI\}_{PKmme^{-1}}}{UE \models MME \sim NI},$$

$$\frac{MME \models (PKue \xrightarrow{UE}) MME \triangleleft \{RES\}_{PKue^{-1}}}{MME \models UE \sim RES} \text{ To satisfy } \mathbf{G3}. \text{ For other phases (i.e., REPS-AKA4Ph.1.2,}$$

REPS-AKA4Ph.2.1, REPS-AKA4Ph.2.2.1, REPS-AKA4Ph.2.2.2, REPS-AKA4Ph.3.1, REPS-AKA4Ph.3.2, REPS-AKA4Ph.4.1, and REPS-AKA4Ph.4.2) we can prove **G3** in the same way. We employ the BAN logic rule (**R3**) to derive:

$$\frac{UE \models \#(RNws), UE \models AP \sim RNws}{UE \models AP \models RNws} \wedge \frac{UE \models WAAA \Rightarrow RNws, UE \models WAAA \models RNws}{UE \models RNws},$$

$$\frac{UE \models \#(RNap2), UE \models AP \sim RNap2}{UE \models AP \models RNap2} \wedge \frac{UE \models WAAA \Rightarrow RNap2, UE \models WAAA \models RNap2}{UE \models RNap2},$$

Thus, the MK and NMK are generated as fresh keys. Hence the protocol satisfies **G4**. Using proofs in **G2** and **G4**, we found that $UE \models AP (WAAA) \models MK (NMK)$ and $AP (WAAA) \models UE \models MK (NMK)$. And by

$$(R2): \frac{UE \models WAAA \Rightarrow MK (NMK), UE \models WAAA \models MK (NMK)}{UE \models MK (NMK)} \text{ Then, we have } UE \xleftrightarrow{MK (NMK)} AP (WAAA) \text{ to}$$

satisfy **G5**.

6.3.4 Proof of security requirements

We present correctness proofs of suggested REPS-AKA4 protocols based on defined security requirements. The security evaluation of the REPS-AKA4 methods are as follows:

Claim 1: The REPS-AKA4 provides resistance to user's identities attacks.

Proof: The REPS-AKA4 protocols protect user's identities in WLAN and LTE networks. In the standard AKA protocol, the user's identity (i.e., IMSI) is transmitted during AKA procedure in clear text. Thus, an LTE user can be found and tracked by means of its captured IMSI. The REPS-AKA4 methods achieve considerable privacy by employing dynamic user's identities (i.e., UI, and NUI, LUI). Also, the methods encrypt IMSI using the public key of APs and eNodeBs before transmissions.

Claim 2: The protocol ensures strong message authentication.

Proof: The REPS-AKA4 protocols can provide message authentication between the involved nodes. For example, in message M3 of REPS-AKA4 ph1.1, the NI value is encrypted by the inverse key of the HSS. Thus, the MME can be sure that the received messages are from valid HSS nodes. We also have the same situation in other REPS-AKA4 protocols.

Claim 3: The REPS-AKA4 achieves protection of AVs and Keys.

Proof: In REPS-AKA4 methods, the AVs are transmitted securely since nodes encrypt AVs using secret and public keys (i.e, SK_{mh}, PK_{ue}, and PK_{ls}). Thus, an attacker cannot be authenticated successfully as a valid node in the system. Since, attacker needs to extract AVs for a successful authentication and extraction of these AVs is impossible in REPS-AKA4 methods.

Claim 4: The REPS-AKA4 achieves resistance to MitM attacks.

Proof: The REPS-AKA4 methods are resistant to the MitM attacks properly. A malicious node cannot extract security parameters such as UI, NUI, NI, and MK since all transitions are encrypted. In REPS-AKA4 protocols, it is not feasible for an attacker to capture the transmitted parameters. Since attackers cannot generate corresponding keys without the main keys (e.g, UKG, UPK, and NK). Also, the UE's identity (i.e. IMSI) is protected by encryption. Thus, the attacker never finds the user's identity and fails to run MitM attacks.

Claim 5: The REPS-AKA4 protocol achieves optimization of Key computation.

Proof: In the re-authentication phases of the REPS-AKA4, the system does not need to generate all keys as the first attachment. It can update keys in a secure manner to optimize cost of key generation. The UPK in REPS-AKA4 ph1.2 and UK in REPS-AKA4 ph2.2.2 are two updating keys which help to decrease key generation overload.

Claim 6: The protocol achieves protection of network identity.

Proof: The HSS authenticates the serving MME via the HNID parameter. Thus, the HSS protects the identity of the MME from being impersonated. The HNID is constructed from MVP and HVP which are shared between HSS and MME.

Claim 7: The proposed REPS-AKA4 provides resistance to Single-key issues.

Proof: The REPS-AKA4 protocols ensure resistance to this security problem. The proposed approach employs a dynamic key generation scheme instead of the standard method. In REPS-AKA4 protocols, UKG, NK, UK, and UPK keys are linked with original key (i.e., SK), RNmme, RNls, RNws, NUI, LUI, LNI, and NI to make other AKA keys. Thus, the involved nodes generate a fresh and secure key for AKA procedure and an attacker cannot obtain any User's data.

Claim 8: The REPS-AKA4 provides robust mutual authentication.

Proof: The REPS-AKA4 schemes provide robust authentication between involved nodes. The UE and HSS can verify each other using NI, NUI, NNI, UNI, UNNI,

AUTN, and UI. The MME authenticates UE by comparing RES, XRES, NRES, and NXRES. The REPS-AKA4 computes these parameters using new calculations and random numbers to increase the security of the transmitted values.

Claim 9: The REPS-AKA4 can increase level of Confidentiality.

Proof: The proposed REPS-AKA4 approaches provide a high level of confidentiality to all LWI interfaces in involved nodes (i.e., UE, MME, eNodeB, WAAA, AP, LAAA, and HSS). Therefore, this method can prevent malicious attacks such as DoS and MitM attacks, which occur because of unprotected transmissions. In ordinary AKA schemes, attackers can extract the transmitted messages in LTE and WLAN systems. The REPS-AKA4 methods are safe under these types of attacks since all transmitted information/parameters are encrypted using secret/public keys as presented in Table 29 (Asym: Asymmetric, Sym: Symmetric).

Table 29: Asymmetric and symmetric encryption in REPS-AKA4

Phase name	Messages and Encryption type
REPS-AKA4 ph1.1	Asym: M1, M4, M5 and Sym: M2, M3
REPS-AKA4 ph1.2	Asym: Re-M1, Re-M3, Re-M4 and Sym: Re-M2
REPS-AKA4 ph2.1	Asym: M1, M4, M5, M8, M9, M12 and Sym: M2, M3, M6,M7,M10,M11
REPS-AKA4 ph2.2.1	Asym: M1, M4, M5, M8, M9, M12 and Sym: M2, M3,M6,M7,M10, M11
REPS-AKA4 ph2.2.2	Asym: M1, M6, M7, M10 and Sym: M2, M3, M4, M5, M8, M9
REPS-AKA4 ph3.1	Asym: M1, M4, M5, M8 and Sym: M2, M3, M6, M7
REPS-AKA4 ph3.2	Asym: M1, M4, M5, M8 and Sym: M2, M3, M6, M7
REPS-AKA4 ph4.1	Asym: M1, M6, M7, M10 and Sym: M2, M3, M4, M5, M8, M9
REPS-AKA4 ph4.2	Asym: M1, M6, M7, M10 and Sym: M2, M3, M4, M5, M8, M9

6.4 Performance analysis of the proposed model: REPS-AKA4

We compare the REPS-AKA4 protocols with other suggested protocols, including standard EPS-AKA. This section is divided into two main subsections (i.e., Evaluation of LTE-AKA methods, and Evaluation of LWI methods).

6.4.1 Evaluation of LTE-AKA methods (REPS-AKA4 ph.1.1/2)

When a UE joins to an LTE network, the UE employs an AKA protocol for authentication and key management. Recall that, REPS-AKA4.ph1.1 and REPS-AKA4.ph1.2 methods are proposed for first time attachment, and n^{th} time attachment respectively. In the following section, we evaluate these phases of REPS-AKA4 based on important security metrics.

6.4.1.1 Bandwidth consumption (core traffic)

We evaluate bandwidth consumption between MME and HSS using a basic mobility model. In this model, the cost of traffic stream is calculated according to Mean Density (MD) of UEs in a cell, the speed (V) of the users, and length of the registration area per cell (L). The assumption values for this model are presented in Table 30. The UE's can move in all directions $[0-2\pi]$. The Rate of authentication request (R) is considered by (6.1) [12, 24, 41] as: $R = (MD \times V \times L)/\pi$ (6.1)

Table 30: The assumption values

Parameters	values (Assumption)
Ar	100
L	25.3 km
MD	350 / km ²
V	4.8 km/h

Therefore, the Total Requests (TR) is expressed as: $TR = R \times Ar$ (6.2)

Where Ar is the number of registration areas. Hence, according to the Table 30, TR value can be found as 376. The bandwidth consumption between MME and HSS (BW_{mh}) can be calculated as [12, 24, 41]: $BW_{mh} = \sum_{k=1}^{k=n} (MSG)_k \times TR$ (6.3)

Where “ n ” is the number of transmitted messages during the AKA procedure, and $(MSG)_k$ is the number of bits in the k^{th} message.

The BW_{mh} of REPS-AKA4.ph1.1: In the suggested REPS-AKA4.ph1.1 protocol, messages shared between the MME and the HSS nodes are: message $MSG1$ and message $MSG2$. The MME uses $MSG1$ to request AKA parameters/keys from the HSS node in home network. And the HSS forwards related AKA parameters/keys to the MME using $MSG2$: $MSG1 = (IMSI) + (RNue) + (RNmme) + (HNID) + (UI) = 128 + 128 + 128 + 128 + 128 = 640 \text{ bits}$, $MSG2 = (NI) + (RNhss) + (RAND) + (AUTH) + (K_{ASME}) + (XRES) + (RNmme) = 128 + 128 + 128 + 128 + 256 + 32 + 128 = 928 \text{ bits}$, $BW_{mh} = (640 + 928) \times TR$.

Based on (6.3) and Table 30 (i.e., $TR=376$), the BW_{mh} of the REPS-AKA4.ph1.1 protocol can be found as 0.58 Mbits/s.

The BW_{mh} of REPS-AKA4.ph1.2: In the REPS-AKA4.ph1.2 that is for re-authentication procedure, message shared between serving and home networks (i.e., MME and HSS) is $MSG1$. The MME forwards Re-M2 (i.e., $MSG1$) message including $RNmme$, $RNue$, $UAUTN$, and $URAND$ parameters to the HSS: $MSG1 = (RNue) + (RNmme) + (UAUTN) + (URAND) = 128 + 128 + 128 + 128 + 128 = 640 \text{ bits}$, $BW_{mh} = (640) \times TR$. Consequently, using (6.3) with the parameters in Table 30 (i.e., $TR=376$), the overall BW_{mh} in the EPC can be considered as 0.24 Mbits/s for REPS-AKA4.ph1.2 (re-authentication phase).

Accordingly, we compare the REPS-AKA4.ph.1.1 and REPS-AKA4.ph.1.2 protocols with MLTE-AKA and the standard EPS-AKA methods based on BW_{mh} .

Comparison of mentioned AKA protocols is shown in Figure 64 for different AV sizes. The evaluation results show that the achieved average improvements of the REPS-AKA4.ph1.1 protocol over the EPS-AKA protocol and MLTE-AKA are 78% and 84% respectively. The achieved average improvements of the REPS-AKA4.ph1.2 protocol over the EPS-AKA protocol and MLTE-AKA are 90% and 93% respectively.

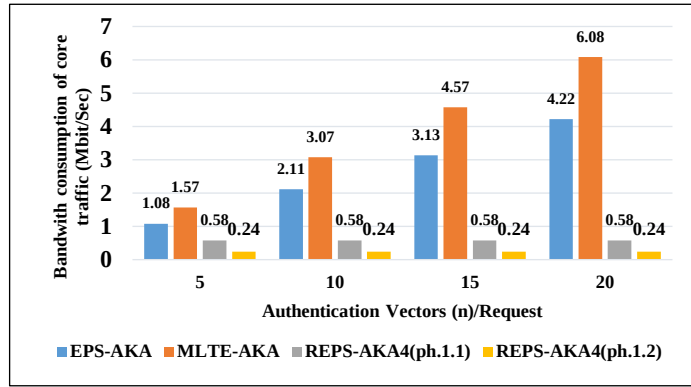


Figure 64: Bandwidth consumption of core traffic

We evaluate the proposed REPS-AKA4.ph1.1 and REPS-AKA4.ph1.2 and other AKA methods according to the MD . Figure 65 illustrates the performance evaluation of the AKA protocols based on different MD values. As expected, BW_{mh} of all protocols increase as MD increases, however, suggested protocols outperform the other methods as they have the lowest bandwidth consumption for core traffic compare with others.

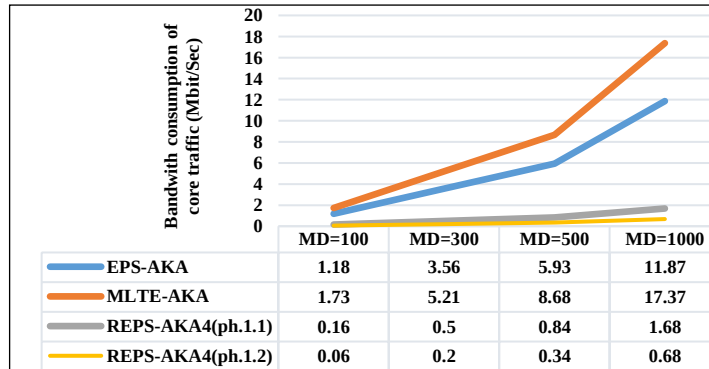


Figure 65: BW_{mh} of the AKA protocols (AV=20)

According to evaluation results, the achieved average improvements of the REPS-AKA4.ph1.1 protocol over the EPS-AKA and MLTE-AKA protocols are 85% and 90% respectively. The achieved average improvements of the REPS-AKA4.ph1.2 protocol over the EPS-AKA and MLTE-AKA are 94% and 96% respectively.

6.4.1.2 Bandwidth consumption (total traffic)

We compare the Total Bandwidth Consumption (*TBC*) of the REPS-AKA4.ph1.1 and REPS-AKA4.ph1.2 with some other AKA methods [1] [12] [58]. We assume that “*n*” number of AVs are sent for each user. The evaluation results are shown in Figure 66. We calculate *TBC* of the protocols according to different numbers of AV (i.e., 5, 10, 15, and 20) with the assumption values in Table 30. The *TBC* of the suggested methods are fixed, conversely, there is an extensive increase in *TBC* of other methods.

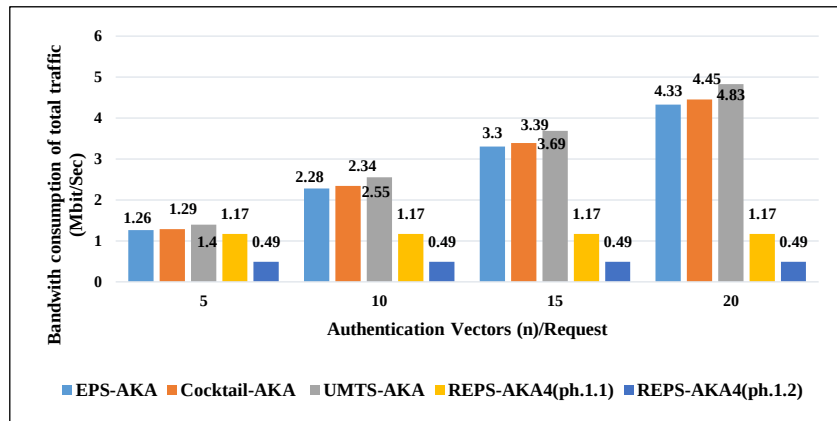


Figure 66: Bandwidth consumption of total traffic

According to evaluation results in Figure 66, the achieved average improvements of the REPS-AKA4.ph1.1 protocol over the EPS-AKA, Cocktail-AKA, and UMTS-AKA are 58%, 59%, and 62% respectively. In addition, the achieved average improvements of the REPS-AKA4.ph1.2 protocol over the EPS-AKA, Cocktail-AKA, and UMTS-AKA are 82%, 83%, and 84% respectively.

We evaluate the AKA protocols based on the *MD* of UEs as shown in Figure 67, when the number of AV is 20. As expected, *TBC* of all protocols rises as *MD* increases, but, REPS-AKA4.ph.1.1 and REPS-AKA4.ph.1.2 outperform the other methods as they have the lowest *TBC* and the smallest slope.

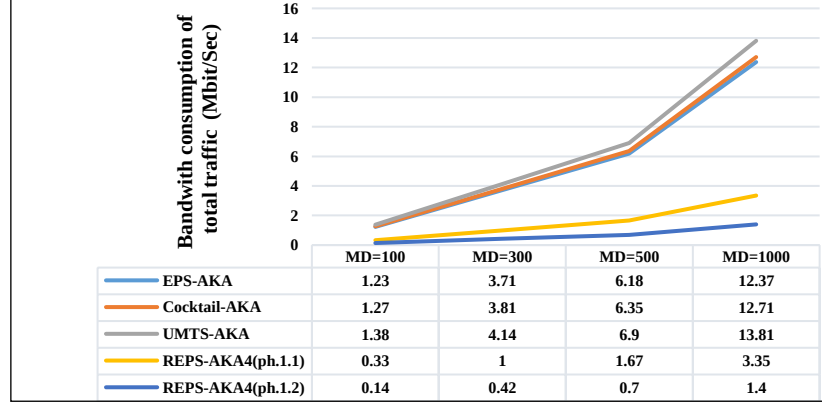


Figure 67: Bandwidth consumption of total traffic (AV=20)

The calculation results show that the achieved average improvements of the REPS-AKA4.ph1.1 protocol over the standard EPS-AKA, Cocktail-AKA, and UMTS-AKA are 73%, 74%, and 76% respectively. The achieved average improvements of the REPS-AKA4.ph1.2 protocol over the standard EPS-AKA, Cocktail-AKA, and UMTS-AKA are 88%, 89%, and 90% respectively.

6.4.1.3 Computational cost

To compute the Computational Cost (*CC*), we evaluate security functions involved in each AKA procedure. The computational cost of each AKA protocol can be expressed by (6.4) [12, 24, 41]:

$$Computation\ Cost = \sum_{k=1}^{k=m} Cost(node)_k \times TR \quad (6.4)$$

Where “*m*” is the number of involved nodes in AKA method.

The CC of REPS-AKA4.ph.1.1: The CC of security functions in the REPS-AKA4.ph.1.1 protocol are as follows: $UE_{cost} = (UIFunc + NIGFunc + NKGFunc + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc) = 10$, $MME_{cost} = (NIFunc) = 1$, $HSS_{cost} = (UIFunc + NIGFunc + NKGFunc + f1 + f2 + f3 + f4 + f5 + KDF + AUTHFunc + NIFunc) = 11$. Hence, using (6.4) the total computation cost of the REPS-AKA4.ph.1.1 protocol can be found as: **Computation Cost (REPS – AKA4.ph.1.1) = $TR \times (22)$** .

The CC of REPS-AKA4.ph.1.2: The CC of security functions in the REPS-AKA4.ph.1.2 protocol are as follows: $UE_{cost} = (NUIFunc + NNIGFunc + UPKFunc + KDF + KDF + KDF + KDF) = 7$, $MME_{cost} = (NUIFunc + UPKFunc + KDF + KDF + KDF + KDF) = 6$, $HSS_{cost} = (NUIFunc + NNIGFunc) = 2$. Hence, using (6.4) the total computation cost of the REPS-AKA4.ph.1.2 protocol can be found as: **Computation Cost (REPS – AKA4.ph.1.2) = $TR \times (15)$** .

According to the Table 30 (TR is 376), the CC (in million) of the REPS-AKA4.ph.1.1 and REPS-AKA4.ph.1.2 can be expressed as 0.008 and 0.005 respectively. Figure 68 displays the overall computational cost of different AKA protocols based on different numbers of AVs. The results show that the suggested REPS-AKA4 methods have less computational cost than standard EPS-AKA and MLTE-AKA protocols.

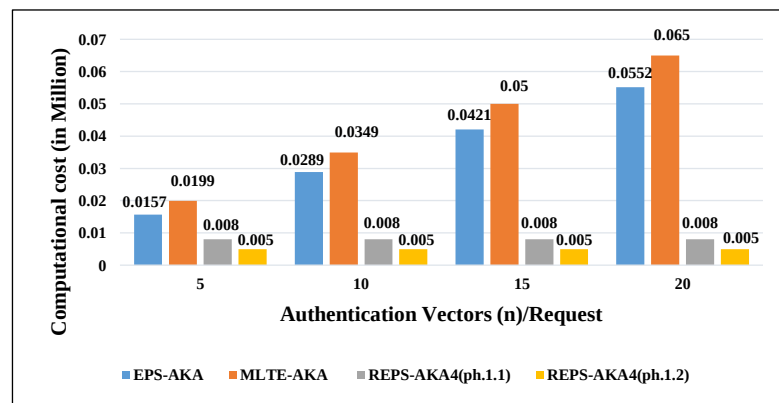


Figure 68: Computational cost of AKA protocols

The average computational cost improvements of REPS-AKA4.ph.1.1 over EPS-AKA and MLTE-AKA are 77% and 80% respectively. Additionally, the average computational cost improvements of REPS-AKA4.ph.1.2 over EPS-AKA and MLTE-AKA are 85% and 88% respectively. The main superiority of our protocols becomes clearer particularly when the number of AV rises. In addition, we evaluate AKA protocols based on MD values. As shown in Figure 69, computational cost of all security protocols rises as *MD* increases. But, REPS-AKA4 protocols outperform the others as they have the lowest computational cost. Based on evaluation results, the achieved average improvements of the REPS-AKA4.ph.1.1 protocol over the EPS-AKA and MLTE-AKA are 85% and 87% respectively. The achieved average improvements of the REPS-AKA4.ph.1.2 protocol over the EPS-AKA and MLTE-AKA are 90% and 92% respectively.

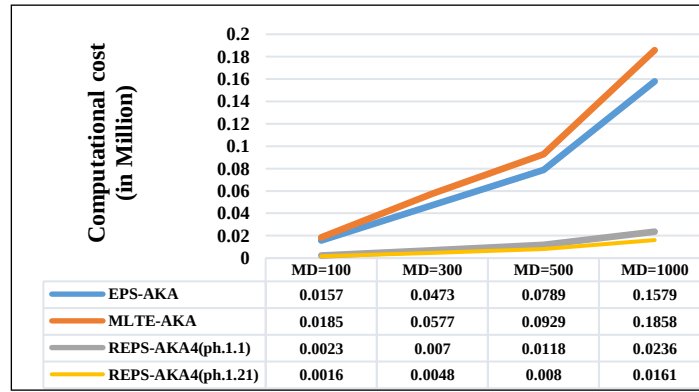


Figure 69: Computational cost in AKA methods (when AV=20)

Moreover, there are some methods which have better computational cost (e.g., Ref. [26]= $20 \times TR$, Ref. [25]= $19 \times TR$). However, according to other performance metrics and security evaluations, the REPS-AKA4 is much better than them.

6.4.2 Evaluation of LWI methods

We evaluate the proposed protocols based on handover authentication scenarios [82]. There are four domains based on UE movement including **Domain1**: when a UE connects to eNodeB of LTE, **Domain2**: when a UE reaches to an area covered by both eNodeB and AP1, **Domain3**: when a UE moves to next area and connects to another AP2 which is connected to the same WAAA server, and **Domain4**: when a UE moves to next area and connects to other AP3 which is connected to different WAAA. The UE initially connects to LTE network (i.e., domain 1), and then moves to WLAN1 (domain 2), WLAN2 (domain 3), and WLAN3 (domain 4). We have considered six different scenarios. In these scenarios, the UE uses different combination of AKA protocols in its journey between eNodeB and APs as shown Figure 70.

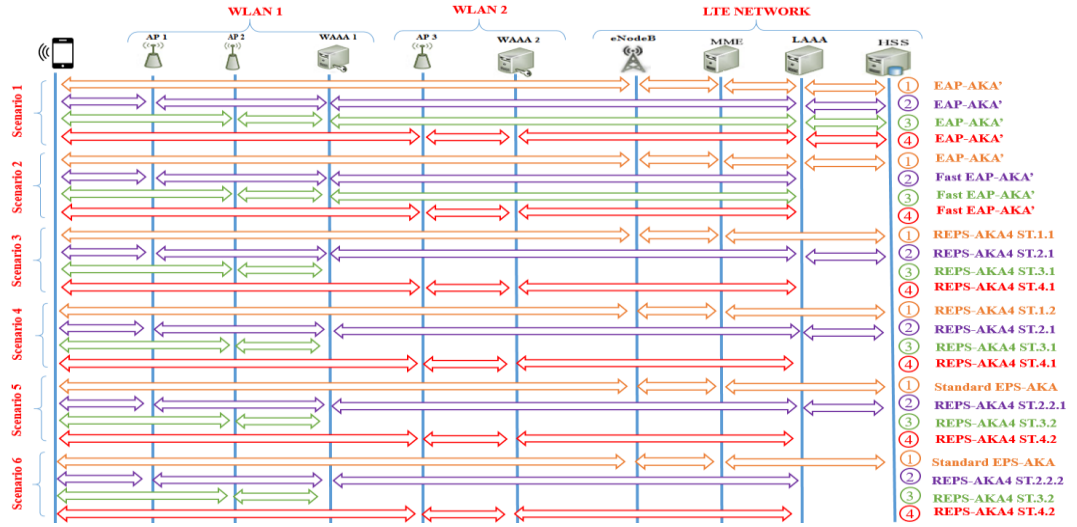


Figure 70: Evaluated scenarios (i.e., 1-6)

In the 1st scenario, we employ four full EAP-AKA' methods to perform the AKA procedures. There are one Full EAP-AKA' and three Fast re-authentication protocols in the 2nd scenario. In the 3rd scenario, the UE uses REPS-AKA4 Ph1.1 protocol for its attachment to LTE network (domain 1), then it employs REPS-AKA4 Ph2.1, REPS-

AKA4 Ph3.1, and REPS-AKA4 Ph4.1 to connect to AP1 (domain 2), AP2 (domain 3), and AP3 (domain 4) respectively. In the 4th scenario, the UE uses REPS-AKA4 Ph1.2 protocol for its attachment to LTE network (domain 1) since the UE connects to the eNodeB for nth time, then it employs REPS-AKA4 Ph2.1, REPS-AKA4 Ph3.1, and REPS-AKA4 Ph4.1 to connect to AP1 (domain 2), AP2 (domain 3), and AP3 (domain 4) respectively. In the 5th scenario, the UE uses standard LTE-AKA protocol for its attachment to LTE network (domain 1), then it employs REPS-AKA4 Ph2.2.1, REPS-AKA4 Ph3.2, and REPS-AKA4 Ph4.2 to connect to AP1 (domain 2), AP2 (domain 3), and AP3 (domain 4) respectively. In the 6th scenario, the UE uses standard LTE-AKA protocol for its attachment to LTE network (domain 1), then it employs REPS-AKA4 Ph2.2.2, REPS-AKA4 Ph3.2, and REPS-AKA4 Ph4.2 to connect to AP1 (domain 2), AP2 (domain 3), and AP3 (domain 4) respectively. We evaluate authentication delay and signaling cost metrics in these scenarios to analyze the performance of proposed REPS-AKA4 methods.

6.4.2.1 Authentication delay

An important evaluation metrics of LWI systems is Authentication Delay (*AD*) which effects handover delay. The *AD* is composed of three components as shown in (6.5)

$$[82]; \quad AD_{total} = D(proc) + D(trans) + D(prop) \quad (6.5)$$

Where $D_{(proc)}$, $D_{(trans)}$, and $D_{(prop)}$ are processing delay, transmission delay, and propagation delay respectively. Total Authentication Delay for each AKA protocol ($TAD_{(auth. protocol)}$) in LWI systems can be composed as follows:

$$\begin{aligned} TAD(auth. protocol) = & AM(ws) \times HC(ws) \times (2D(proc.ws) + D(trans.ws) + \\ & D(prop.ws)) + AM(wd) \times HC(wd) \times (2D(proc.wd) + D(trans.wd) + \\ & D(prop.wd)) + DR(av) + D(4HS) \end{aligned} \quad (6.6)$$

Where $AM_{(ws)}$ and $AM_{(wd)}$ are the number of transmitted messages in the wireless networks and wired networks, respectively. The $HC_{(ws)}$ and $HC_{(wd)}$ values are number of hops between wireless and wired entities, respectively. The $D_{(trans.ws)}$ is defined as the transmission delay in the wireless section of the LWI system, and the $D_{(trans.wd)}$ is the transmission delay in the wired section of the LWI system. The processing delay in wireless and wired sections of the LWI system are defined by the $D_{(proc.ws)}$ and $D_{(proc.wd)}$ respectively. The $D_{(prop.ws)}$ is the propagation delay of the wireless section of LWI system, and the $D_{(prop.wd)}$ is the propagation delay of the wired part of the system. The $DR_{(av)}$ is the delay of generating the AV in the HSS. And the $D_{(4HS)}$ is the delay of the four-way handshake process. In our evaluations, we assume that Hop Count (HC) between WAAA server and LAAA server is 7 (i.e., $HC=7$), and there is one hop (i.e., $HC=1$) between each other entity. The $TAD_{(auth. protocol)}$ for standard Full authentication (EAP-AKA'), Fast re-authentication (fast EAP-AKA') and EPS-AKA protocols can be calculated as follows [82]:

$$TAD(auth. EAP - AKA') = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 34(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS),$$

$$TAD(auth. Fast EAP - AKA') = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 32(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS),$$

$$TAD(auth. Standard EPS - AKA') = 4(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 5(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS).$$

Additionally, the $TAD_{(auth. protocol)}$ for the proposed REPS-AKA4 protocols can be calculated as follows:

$$TAD(auth. REPS - AKA4 Ph1.1) = 4(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 5(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS),$$

$$TAD(auth. REPS - AKA4 Ph1.2) = 4(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 4(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS),$$

$$TAD(auth. REPS - AKA4 Ph2.1) = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 20(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS),$$

$$TAD(auth. REPS - AKA4 Ph2.2.1) = 5(2D(proc.ws) +$$

$$\begin{aligned}
& D(trans.ws) + D(prop.ws)) + 20(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + \\
& D(4HS), TAD(auth.REPS - AKA4 Ph2.2) = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + \\
& 18(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS), TAD(auth.REPS - \\
& AKA4 Ph3.1) = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 4(2D(proc.wd) + D(trans.wd) + \\
& D(prop.wd)) + DR(av) + D(4HS), TAD(auth.REPS - AKA4 Ph3.2) = 5(2D(proc.ws) + \\
& D(trans.ws) + D(prop.ws)) + 4(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + \\
& D(4HS), TAD(auth.REPS - AKA4 Ph4.1) = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + \\
& 18(2D(proc.wd) + D(trans.wd) + D(prop.wd)) + DR(av) + D(4HS), TAD(auth.REPS - \\
& AKA4 Ph4.2) = 5(2D(proc.ws) + D(trans.ws) + D(prop.ws)) + 18(2D(proc.wd) + D(trans.wd) + \\
& D(prop.wd)) + DR(av) + D(4HS).
\end{aligned}$$

When using the standard Full/Fast EAP-AKA', standard EPS-AKA, and the suggested REPS-AKA4 methods in the user movements model as shown in Figure 56, the Total Delay of each scenario ($TD_{scenario}$) can be presented as follows: $TD_{scenario 1} = 4 \times$

$$\begin{aligned}
& TAD(auth.EAP - AKA'), TD_{scenario 2} = TAD(auth.EAP - AKA') + 3 \times TAD(auth.Fast EAP - \\
& AKA'), TD_{scenario 3} = TAD(auth.REPS - AKA4 Ph1.1) + TAD(auth.REPS - AKA4 Ph2.1) + \\
& TAD(auth.REPS - AKA4 Ph3.1) + TAD(auth.REPS - AKA4 Ph4.1), TD_{scenario 4} = \\
& TAD(auth.REPS - AKA4 Ph1.2) + TAD(auth.REPS - AKA4 Ph2.1) + TAD(auth.REPS - AKA4 Ph3.1) + \\
& TAD(auth.REPS - AKA4 Ph4.1), TD_{scenario 5} = TAD(auth.Standard EPS - AKA) + \\
& TAD(auth.REPS - AKA4 Ph2.2.1) + TAD(auth.REPS - AKA4 Ph3.2) + TAD(auth.REPS - \\
& AKA4 Ph4.2), TD_{scenario 6} = TAD(auth.Standard EPS - AKA) + TAD(auth.REPS - \\
& AKA4 Ph2.2.2) + TAD(auth.REPS - AKA4 Ph3.2) + TAD(auth.REPS - AKA4 Ph4.2).
\end{aligned}$$

We use the values of the mentioned parameters in (6) based on [96] and [97]. Accordingly, the $D_{(trans.ws)}$ and $D_{(trans.wd)}$ are set to 2 ms and 0.5 ms respectively. The $D_{(proc.ws)}$ and $D_{(proc.wd)}$ are set to 0.00 ms, and 0.00 ms respectively. The value of the $D_{(4HS)}$ can be ignored in the calculation, because its effect in each scenario is same. The $DR_{(av)}$ is fixed to 0.001 ms [82]. Thus, the performance of scenarios can be appraised based on AD for various D_{prop} (0.05-0.5 ms) when the sampling interval is 0.05 ms. As can be seen in Table 31, the $TD_{scenario}$ rises when the D_{prop} is increased.

Figure 71 illustrates the comparison of average authentication delay for the mentioned scenarios. The evaluation results show that the performance of scenarios 3, 4, 5, and 6 which use REPS-AKA4 protocols outperform the scenarios 1 and 2 which used Standard Full EAP-AKA' and Fast EAP-AKA'.

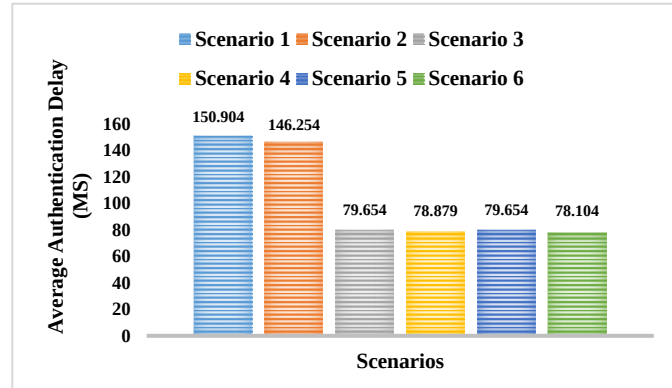


Figure 71: Comparison of average authentication delay

The evaluation results indicate that the achieved average improvements of the scenarios 3, 4, 5, and 6 (which use REPS-AKA4 protocols) over the scenarios 1, and 2 (which use standard AKA protocols) are 47% and 45% respectively.

Table 31: Authentication delay in different scenarios

Propagation Delay (ms)	Authentication Delay (ms)					
	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
0.05	115.804	112.504	64.804	64.254	64.804	63.704
0.1	123.604	120.004	68.104	67.504	68.104	66.904
0.15	131.404	127.504	71.404	70.754	71.404	70.104
0.2	139.204	135.004	74.704	74.004	74.704	73.304
0.25	147.004	142.504	78.004	77.254	78.004	76.504
0.3	154.804	150.004	81.304	80.504	81.304	79.704
0.35	162.604	157.504	84.604	83.754	84.604	82.904
0.4	170.404	165.004	87.904	87.004	87.904	86.104
0.45	178.204	172.504	91.204	90.254	91.204	89.304
0.5	186.004	180.004	94.504	93.504	94.504	92.504

6.4.2.2 Signaling cost

Signaling Cost (SC) is one of the most significant measurements to analyze the performance of the AKA protocols. The SC for each scenario can be measured as follows [82]:

$$SC(\text{scenario}) = N(\text{auth.protocol}) \times \text{Message}(\text{Size}) \times AN(UM) \quad (6.7)$$

The $N(\text{auth.protocol})$ is the number of transmitted messages during an AKA protocol. The $\text{Message}(\text{Size})$ is message size of transmitted packets. Thus, the SC for scenarios 1-6 are as follows:

$$\begin{aligned} SC(\text{scenario 1}) &= (4 \times N(\text{auth.EAP} - \text{AKA}')) \times \text{Message}(\text{Size}) \times AN(UM), \\ SC(\text{scenario 2}) &= (1 \times N(\text{auth.EAP} - \text{AKA}') + 3 \times N(\text{auth.Fast EAP} - \text{AKA}')) \times \text{Message}(\text{Size}) \times AN(UM), \\ SC(\text{scenario 3}) &= (N(\text{auth.REPS} - \text{AKA4 Ph1.1}) + N(\text{auth.REPS} - \text{AKA4 Ph2.1}) + N(\text{auth.REPS} - \text{AKA4 Ph3.1}) + N(\text{auth.REPS} - \text{AKA4 Ph4.1})) \times \text{Message}(\text{Size}) \times AN(UM), \\ SC(\text{scenario 4}) &= (N(\text{auth.REPS} - \text{AKA4 Ph1.2}) + N(\text{auth.REPS} - \text{AKA4 Ph2.1}) + N(\text{auth.REPS} - \text{AKA4 Ph3.1}) + N(\text{auth.REPS} - \text{AKA4 Ph4.1})) \times \text{Message}(\text{Size}) \times AN(UM), \\ SC(\text{scenario 5}) &= (N(\text{auth.Standard EPS} - \text{AKA}) + N(\text{auth.REPS} - \text{AKA4 Ph2.2.1}) + N(\text{auth.REPS} - \text{AKA4 Ph3.2}) + N(\text{auth.REPS} - \text{AKA4 Ph4.2})) \times \text{Message}(\text{Size}) \times AN(UM), \\ SC(\text{scenario 6}) &= (N(\text{auth.Standard EPS} - \text{AKA}) + N(\text{auth.REPS} - \text{AKA4 Ph2.2.2}) + N(\text{auth.REPS} - \text{AKA4 Ph3.2}) + N(\text{auth.REPS} - \text{AKA4 Ph4.2})) \times \text{Message}(\text{Size}) \times AN(UM) \end{aligned}$$

The Number of Transmitted Messages (NTM) and Average Message Size (AMS) of the Full EAP-AKA', Fast EAP-AKA', EPS-AKA, REPS-AKA4 (all phases) are illustrated in Table 32 (when $HC_{(wd)} = 7$).

Table 32: NTM and AMS values of AKA protocols

Protocols name	NTM	AMS (bytes)	Protocols name	NTM	AMS(bytes)
Full EAP-AKA'	39	100	REPS-AKA4 Ph2.2.1	25	60
Fast EAP-AKA'	37	100	REPS-AKA4 Ph2.2.2	23	45
Standard EPS-AKA	9	100	REPS-AKA4 Ph3.1	9	40
REPS-AKA4 Ph1.1	9	70	REPS-AKA4 Ph3.2	9	40
REPS-AKA4 Ph1.2	8	50	REPS-AKA4 Ph4.1	23	40
REPS-AKA4 Ph2.1	25	50	REPS-AKA4 Ph4.2	23	40

The average number of UE movements $AN_{(UM)}$ can be calculated as follow:

$$AN(UM) = ST/RT \quad (6.8)$$

Where the ST is the average session time. In our model, we set the ST to 1800 s (i.e., 30 minutes). The RT is the average time that the UE resides in LWI network. We assume that the RT varies between 10 and 100 s. Figure 56 shows the SC (in KB) for the presented scenarios according to the mentioned resident time of the UE (i.e., $RT=(10ms,100ms)$). As can be seen in Figure 72, when the RT of the UE is increased, the SC is decreased. The scenarios (i.e., scenarios 3, 4, 5, and 6) which use the proposed protocols have less SC compared with other scenarios (i.e., scenarios 1 and 2) which used the other protocols.

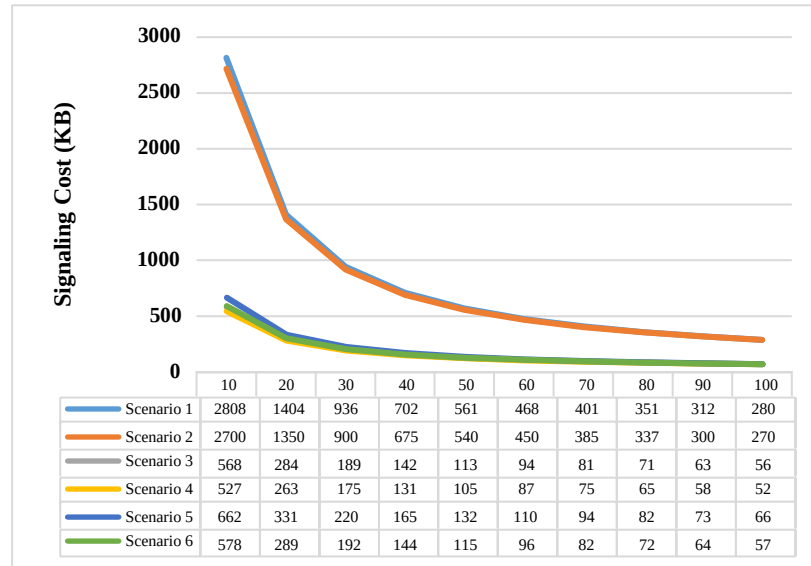


Figure 72: Comparison of signaling cost

Figure 73 shows the comparison of average SC for scenarios 1-6. As can be seen in Figure 73, the average SC of the scenarios 3, 4, 5, and 6 which use REPS-AKA4 protocols are less than the scenarios 1 and 2 which use standard methods.

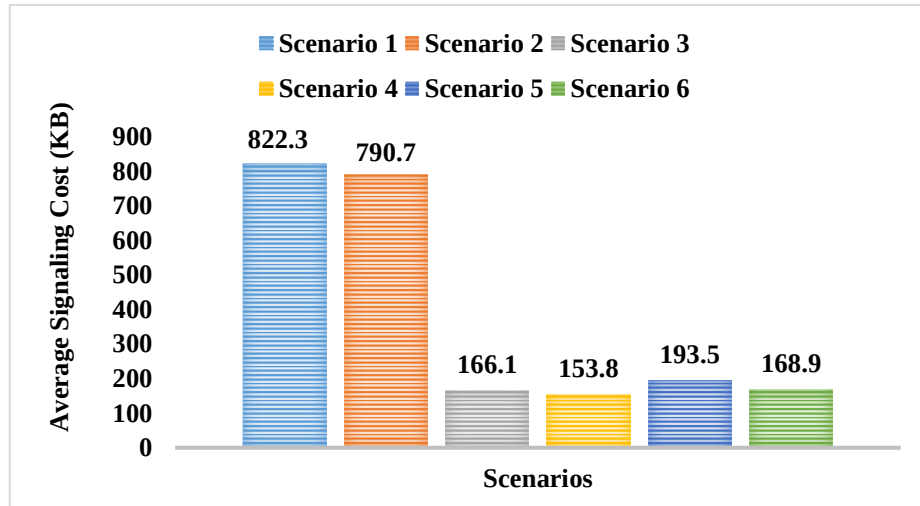


Figure 73: Comparison of average signaling cost

The calculation results indicate that the achieved average improvements of the scenarios 3, 4, 5, and 6 (which use REPS-AKA4 protocols) over the scenarios 1, and 2 (which use standard AKA protocols) are 79% and 78% respectively.

Chapter 7

MODEL 5 (REPS-AKA5)

The Internet of Things (IoT) is a system of interrelated electronic and mechanical devices named “Things” that have specified identifiers and the ability to exchange and share required data in communication networks automatically and without Human to Computer (H2C) or Human to Human (H2H) interaction. In the IoT systems, we utilize Machine to Machine (M2M) communication or Machine Type Communications (MTCs). The M2M system and its applications such as smart city, smart home, and smart grid have been growing rapidly in recent times [9] [38]. One of the most important usage of M2M communication is to make a continual connectivity between IoT Devices (IoTDs). In the IoT system, the IoTDs employ MTC technology to communicate with each other without H2H/H2C interference [98]. The IoT systems have some important features such as low power of IoTDs, real-time requirements, and no user intervention. Thus, the MTC technologies have been developed considerably to manage plenty of real time IoT system (i.e., applications). The most significant applications of IoT system are smart transportation, automatic monitoring systems, smart city, Building Management Systems (BMS), smart grid, and smart health care system [1] [49] [69].

There has been an overwhelming growth in wireless communication systems during the last decade. An outstanding achievement of the M2M communication systems is that not only humans but also objects (i.e., Things in IoT systems) can contact each

other by using the different communication systems. The IoTDs use some instruments (i.e., different sensors) for information gathering [42] [100]. These devices are used for collecting required information and transmitting them to the IoT applications (i.e., IoT servers). The IoT systems employ communication technologies such as WiFi, LTE, and ZigBee to share IoT data between involved nodes. The IoT servers convert received data into the meaningful information. Then, the IoT system makes its decision accordingly [9]. LTE systems play an outstanding role in IoT applications because of some main features including low transmission latency, high data rate, suitable bandwidth/capacity, and proper flexibility. The MTC technology is standardized by 3rd Generation Partnership Project (3GPP) and considerably practiced in industrial environments [101] [9]. Different from the common LTE devices, LTE-enabled IoTDs need some defined requirements (e.g., lower power consumption). To handle MTC, the 3GPP network (i.e., LTE system) needs to handle a huge number of MTC devices (i.e., IoTDs) properly. Hence, LTE system requires managing network resources appropriately to avoid congestion issues at both the data and control plans [102].

There are some security requirements for M2M communication specified by the 3GPP in Release-11 [42]. When a huge number of LTE-enabled IoTDs need to connect to the network simultaneously, each node establishes a full standardized authentication protocol named EPS-AKA. These Authentication and Key Agreement (AKA) procedures can generate a considerable signaling overload over the LTE network. A high signaling overload causes performance-based issues for IoT applications [9]. For instance, when different groups of IoTDs detect an eNodeB (i.e., LTE base station), all these IoT nodes transmit their attach requests toward the LTE network simultaneously. These huge transmissions lead to considerable overload and signaling

congestion in the involved LTE nodes. The standard EPS-AKA approach does not overcome the performance-based issues (i.e., bandwidth overhead and signaling congestion) over the LTE-enabled IoT systems. In addition, the wireless channels between the IoTDs and the LTE networks make possibilities of security threats including eavesdropping/manipulation of user's data, MitM attacks, and DoS attacks [2] [3] [11] [25] [39] [41]. Thus, a secure and congestion avoidance AKA approach for LTE-enabled IoT systems is required. Authentication and privacy of IoTDs are the two principal issues in the LTE-enabled IoT systems [103][104][105]. In addition, the IoTDs such as different sensors and small electronic devices usually have low and limited power resources. If involved IoTDs are unable to be authenticated properly, they would be unable to connect to the network [42]. Hence, it is a significant need to propose a lightweight protocol for LTE-enabled IoT systems. In this chapter, we have presented a group-based AKA protocol named as REPS-AKA5 by leveraging IoT group leader. The suggested method can verify a huge number of IoTDs in a specified IoT group appropriately without any need for a complete AKA process. The proposed REPS-AKA5 protocol mutually authenticates a group of IoTDs in the LTE-enabled IoT systems. This method generates the secure session keys between IoT Leader and each IoT, and between IoT Leader and network nodes. Accordingly, this approach provides secure communication between each IoT group and the LTE system.

The remaining parts of this chapter are organized as follows: Section 7.1 presents an overview of LTE-enabled IoT systems. In section 7.2, details of the REPS-AKA5 method are described. In section 7.3, we provide a security analysis of the suggested approach by using the AVISPA, Scyther tools and BAN logic. Performance analysis of the protocol is illustrated in section 7.4.

7.1 Architecture of LTE-enabled IoT system

In this section, we present a short overview of the M2M (i.e., MTC) communication considered for the IoT systems [1] [9] [38] [105]. The architecture of the MTC system based on the LTE network is shown in Figure 74.

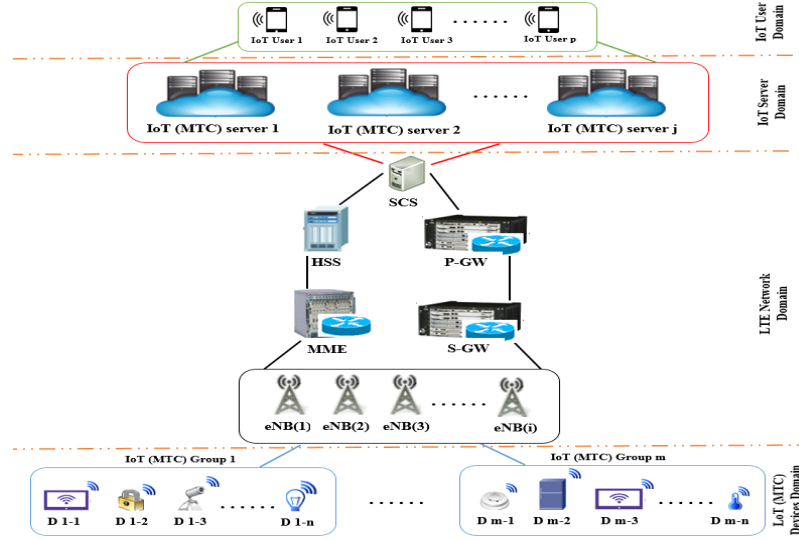


Figure 74: Architecture of LTE-enabled IoT system

The LTE-enabled IoT system can be divided into four main domains including IoT Devices Domain (i.e., IoT Groups), LTE Network Domain, IoT Server Domain (i.e., Application domain), and IoT User Domain [42] [60] [105]:

IoT Devices Domain (Group scope): In this domain, IoTDs form the IoT groups based on some specified parameters such as behavior of the IoTDs, geographical location of each node, and IoT applications. Then, each IoT group selects an IoT Leader (IoTL) among the group members based on IoT application. The IoTDs contact the IoT server (i.e., application) by using the 3GPP network domain (i.e., LTE system). All the IoT groups can be controlled by IoT users from the IoT user domain. To satisfy the security requirements of the IoT system, each IoTDs must be authenticated by the

LTE network. Moreover, when one device is added or removed to/from the IoT group, the state of that group must be updated.

LTE Network Domain: The LTE network is divided into the E-UTRAN and EPC which are Evolved Universal Terrestrial Radio Access Network and the Evolved Packet Core respectively [2] [3] [11] [25] [41]. The E-UTRAN contains some base stations (i.e., eNodeBs) and manages radio channels between the IoTDs and the eNodeBs. Whereas, in the EPC part of the network, there are some nodes such as HSS, MME, S-GW, P-GW, and SCS which are Home Subscriber Server, Mobility Management Entity, Serving Gateway, Packet Gateway, and the Service Capability Server respectively. The HSS works as a central database to store the security parameters of all the LTE users (i.e., IoTDs). The MME manages the related security parameters and AKA procedures. The S-GW and also the P-GW both work in user plans and the SCS is a service provider for IoT applications.

IoT (MTC) Server Domain: In this domain, we have the IoT Servers (IoTS) also known as the IoT applications. An IoTS is a server which has the access to IoTDs by communication systems (i.e., LTE network). Moreover, the IoT users can connect to the IoTS in order to control a group of IoTDs as indicated in Figure 58. The IoTS has access to the IoT user by using an API which is an Application Program Interface. The LTE system provides two kinds of connections to the IoT servers. In the first type, the IoTS is located within the LTE network domain and can be controlled by the network operators. Whereas, in the second type, the IoT servers can be placed outside the LTE network and operators have no control over the IoT servers. In our considered architecture, IoT servers are located outside the LTE network domain.

IoT (MTC) User Domain: In this domain, there are a number of IoT users. The IoT users employ different control devices such as mobiles, tablets, and laptops. Authenticated and authorized IoT users can use all defined services in the IoT systems. The IoT system provides a proper API for IoT users to manage a group of IoT devices.

7.2 The proposed REPS-AKA5 protocol

The proposed REPS-AKA5 method for LTE-enabled IoT systems is illustrated in this section. When “ n ” numbers of IoT devices from “ m ” IoT groups want to be authenticated by the LTE system, some performance based and security issues may occur. To overcome these security issues, we propose a group-based AKA approach to achieve all defined security and performance goals. To identify an IoT device within the IoT system, each IoT device has two connectivity identities including DCID and DTID which are constant and temporary ID respectively. The suppliers install constant identity (i.e., DCID) value over the IoT devices at the time of manufacturing. We use the DCID for the registration over the 3GPP system (i.e., LTE network). The temporary ID (i.e., DTID) is generated during AKA procedure. The n^{th} IoT device of m^{th} group (IoT _{$n-m$}) and the HSS shares a unique secret key SK _{$n-m$} that cannot be sent via the wireless channels. In addition, the IoT Leader of the m^{th} group (IoT_{L m}) and the HSS of network share a Group external Key (GXK _{m}). This key is used to generate Secret Identity of m^{th} Group (SIG _{m}). All the IoT devices, IoT Leaders, MMEs and the HSSs need authentication functions (i.e., F1, F2, F6, F7, F8, and F9) and key generation functions (i.e., F3, F4, F5, and KDF). The HSS and IoT Leaders maintain and update the list of IoT devices in m^{th} group (i.e., GL _{m}). The GL _{m} stores all information of m^{th} IoT group including identities of the IoT devices, secret keys of group member (i.e., SK _{$n-m$}), group external and internal keys (i.e., GXK _{m} , GIK _{m}), and ID of m^{th} Group (i.e., GID _{m}). The suggested protocol includes four phases

including Group Construction Phase, First AKA Phase, Re-authentication Phase, and Group Updating Phase which are explained below.

7.2.1 Group construction phase

To access the LTE network, the Key Generate Center (KGC) assigns a unique identifier to each IoTD (i.e., DCID). The KGC is usually in the LTE network domain and is integrated with the HSS node. When an IoTD in a specified group registers to the LTE system, the KGC produces its needed identifiers and private key. The network nodes (i.e., MME and HSS) have the same function as the IoTDs has. Thus, only the valid nodes can access the private keys which are generated by the KGC. It is also assumed that IoTDs can support both WiFi and LTE networks. The IoTDs belonging to the same IoT application can be configured as one IoT group. The IoT groups can be controlled by one or more IoT users. Moreover, the IoTDs can form several IoT groups according to different grouping methods. The IoT groups are formed based on common criteria of the involved IoTDs like, existing in the same location, having the same behaviors, using the same services, running the same IoT applications, and belonging to the same IoT users. Hence, a set of IoTDs with the same local area can form one or more certain IoT groups. Then, an IoT device with higher storage/capability must be chosen as a trusted group leader (i.e., IoTL). A group leader must be selected based on defined metrics such as computing power of the IoTD, communication capability, and battery/storage status. The selected group leader must be able to implement many AKA processes and provide safe communication. Thus, it must be able to run the security protocols properly. During leader selection, a Group Leader List (GLL) is constructed where IoTDs are sorted according to defined features. The First device in a GLL is the group leader and the second one is a backup leader. The IoTL is responsible for sending/receiving data to/from each IoTD and

handling AKA procedure between LTE network and IoTDLs. The IoTDL generates some security parameters (i.e., $AUTH_{n-m}$ and $SKDL_{n-m}$) to verify all group members. If the group leader does not verify both the IoTDLs and LTE nodes, the AKA process aborts. The secret identity of each group (i.e., SIG_m) must be generated with the participation of all IoTDLs in group leader device. This SIG_m value would be updated after each successful AKA procedure.

7.2.2 First Authentication phase

In the first AKA phase, the IoTDLs and the network nodes carry out the mutual authentication process using the REPS-AKA5 protocol. After a successful AKA procedure, trusted session keys are established between IoTDLs and MME on behalf of the LTE network. Thus, the AKA method provides a secure transmission in the data/control plans. In the method, strong authentication between the IoTDLs and the LTE nodes is executed by using an optimized challenge/response mechanism as shown in Figure 75. The steps of first authentication phase are illustrated as follows:

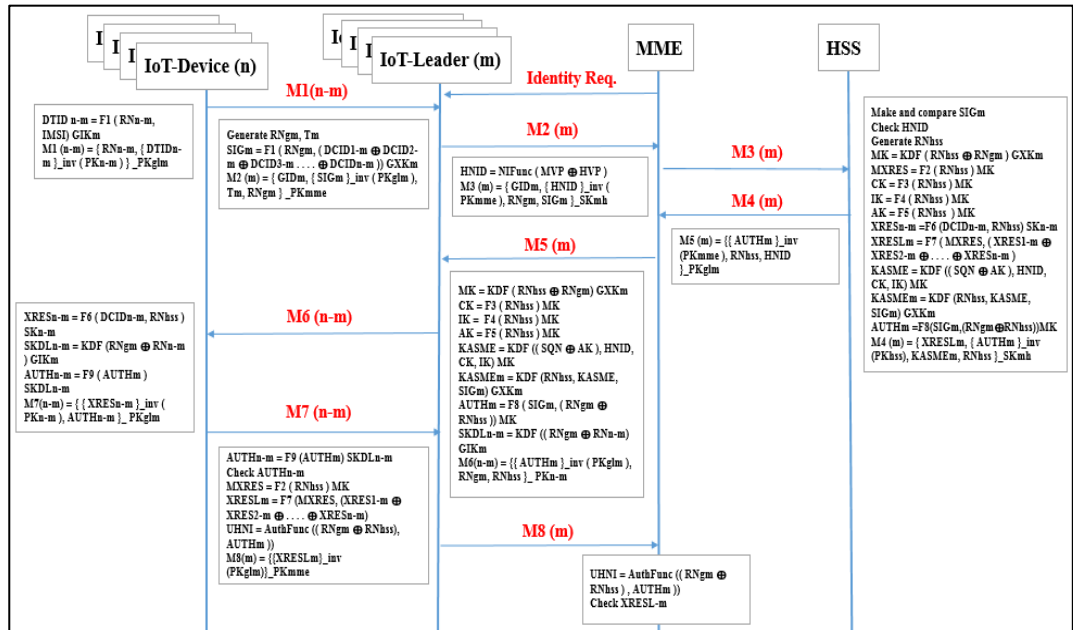


Figure 75: Steps of REPS-AKA5 (First Authentication phase)

1. $\text{IoT}_{n-m} \rightarrow \text{IoTL}_m$: All IoTs of each IoT group send an attach request to the MME node through IoTL_m . Each IoT generates a random number named RN_{n-m} , which means that it is generated by the n^{th} device of the m^{th} group. Then, IoT calculates its own temporary identity (DTID_{n-m}) using IMSI value and RN_{n-m} ; $\text{DTID}_{n-m} = \text{F1}(\text{RN}_{n-m}, \text{IMSI}) \text{GIK}_m$. Each IoT_{n-m} forwards these parameters (RN_{n-m} and DTID_{n-m}) to the IoTL_m via $\text{M1}_{(n-m)}$ message encrypted by the public key of the group leader (PK_{glm}).

2. $\text{IoTL}_m \rightarrow \text{MME}$: When the IoTL_m receives all $\text{M1}_{(n-m)}$ messages from all the IoTs, it extracts received DTID_{n-m} and RN_{n-m} . It also makes/updates the Group List (GL_m) which contains information of all the IoTs in the m^{th} group. Then, the IoTL_m generates a random value named RN_{gm} and a Timestamp (T_m). Then, it computes a secret identity for the group (i.e., SIG_m) using F1 (i.e., Authentication function), DCIDs, RN_{gm} , and the group external key (i.e., G XK_m); $\text{SIG}_m = \text{F1}(\text{RN}_{\text{gm}}, (\text{DCID}_{1-m} \oplus \text{DCID}_{2-m} \oplus \dots \oplus \text{DCID}_{n-m})) \text{G XK}_m$. The IoTL_m encrypts the SIG_m by the inverse public key of group leader. Then, the IoTL_m sends calculated parameters to the MME via $\text{M2}_{(m)}$ encrypted by PK_{mme} (i.e., public key of MME).

3. $\text{MME} \rightarrow \text{HSS}$: After receiving the $\text{M2}_{(m)}$ from IoTL_m , The MME decrypts the received message. Then, this node adds its own identity, and passes the message to the HSS node. The MME generates Home Network Identity (HNID) by using MVP and HVP values: $\text{HNID} = \text{NIFunc}(\text{MVP} \oplus \text{HVP})$. For each IoTL_m (when $m=1, \dots, i$), the MME checks if T_m is expired or valid. If holds, then the MME goes to the next step. Otherwise, it forwards a request failure message to the IoTL_m to notify the invalid T_m . After successful verification of T_m , the MME forwards $\text{M3}_{(m)}$ to the HSS. This message contains $\text{GID}_m, \{\text{HNID}\}_{\text{inv}} (\text{PK}_{\text{mme}}), \text{RN}_{\text{gm}}$, and SIG_m . The $\text{M3}_{(m)}$ is encrypted by a secret key between MME and HSS (i.e., SK_{mh}).

4. HSS → MME: When the HSS receives $M3_{(m)}$ message from serving network, it decrypts and extracts included parameters. Then, the HSS generates its own random number (RN_{hss}) and calculates SIG_m to verify IoT group. If received SIG_m and generated one are found to be equal, the HSS validates the IoT group successfully. After successful verification, the HSS computes Master Key (MK) using RN_{hss} , received RN_{gm} , and GXK_m : $MK = KDF((RN_{hss} \oplus RN_{gm}), GXK_m)$. The HSS uses calculated MK, and two types of functions including message authentication functions (i.e., F1, F2, F6, F7, F8, and F9) and key generation functions (i.e., F3, F4, F5, and KDF) to generate a Group Authentication Vector (GAV_m) for each IoT group. The HSS generates AKA parameters/keys for GAV_m as follows: Main XRES (MXRES); $MXRES = F2(RN_{hss}) MK$, Cipher Key (CK); $CK = F3(RN_{hss}) MK$, Integrity Key (IK); $IK = F4(RN_{hss}) MK$, Anonymity key (AK); $AK = F5(RN_{hss}) MK$, XRES for n^{th} IoT of m^{th} group ($XRES_{n-m}$); $XRES_{n-m} = F6(DCID_{n-m}, RN_{hss}) SK_{n-m}$ where SK_{n-m} is a unique secret key for n^{th} IoT of m^{th} group, XRES for IoT of the m^{th} group ($XRESL_m$); $XRESL_m = F7(MXRES, (XRES_{1-m} \oplus XRES_{2-m} \oplus \dots \oplus XRES_{n-m}))$, $KASME = KDF((SQN \oplus AK), HNID, CK, IK) MK$ where SQN is a Sequence Number, KASME of m^{th} group ($KASME_m$); $KASME_m = KDF(RN_{hss}, KASME, SIG_m) GXK_m$, Authentication Token of the m^{th} group ($AUTH_m$); $AUTH_m = F8(SIG_m, (RN_{gm} \oplus RN_{hss}) MK$. After these calculation, it forms $M4_{(m)}$ message including $XRESL_m$, $\{AUTH_m\}_{inv}(PK_{hss})$, $KASME_m$, RN_{hss} . The $M4_{(m)}$ is encrypted by SK_{mh} and sent to the MME over the secure channel.

5. MME → IoTL_m: The MME decrypts the received message $M4_{(m)}$ using its secret key SK_{mh} and forwards security parameters including $\{AUTH_m\}_{inv}(PK_{mme})$, RN_{hss} , and HNID to the IoTL_m via the $M5_{(m)}$ encrypted by PK_{glm} .

6. $\text{IoTL}_m \rightarrow \text{IoTD}_{n-m}$: After acquiring $M5_{(m)}$ from the MME, the IoTL_m verifies whether the received AUTH_m is equal to the calculated AUTH_m or not. If they match, the IoTL_m authenticates the MME and broadcasts the $M6_{(n-m)}$ to all IoTDs in m^{th} group; otherwise, the IoTL_m declines the authentication process. After successful verification of MME, the IoTL_m computes MK using RN_{hss} , RN_{gm} , and key GXK_m . Then, it generates CK, IK, AK, KASME, and KASME_m same as HSS in home network. Moreover, the IoTL_m calculates a SKDL_{n-m} which is a secret key between IoTL_m and IoTD_{n-m} ; $\text{SKDL}_{n-m} = \text{KDF}(\text{RN}_{\text{gm}} \oplus \text{RN}_{n-m}) \text{ GIK}_m$. The group leader employs the SKDL_{n-m} to produce AUTH_{n-m} value; $\text{AUTH}_{n-m} = \text{F9}(\text{AUTH}_m) \text{ SKDL}_{n-m}$. Then, the IoTL_m forwards $M6_{(m-n)}$ including $\{\text{AUTH}_m\}_{\text{inv}(\text{PK}_{\text{glm}})}$, RN_{hss} , and RN_{gm} to the IoTDs . This messages are encrypted by public key of each IoTD_{n-m} (PK_{n-m}). All IoTDs in each group overhear this broadcasting message $M6_{(n-m)}$.

7. $\text{IoTD}_{n-m} \rightarrow \text{IoTL}_m$: After receiving $M6_{(n-m)}$ message from group leader, each IoTD_{n-m} extracts the included parameters. Then, it produces XRES_{n-m} , and SKDL_{n-m} the same as HSS and IoTL_m respectively. Then, IoTD_{n-m} calculates the value of AUTH_{n-m} using received AUTH_m , and generated SKDL_{n-m} . After that, each IoTD_{n-m} forwards $M7_{(n-m)}$ message to the group leader. The $M7_{(n-m)}$ includes $\{\text{XRES}_{n-m}\}_{\text{inv}(\text{PK}_{n-m})}$ and AUTH_{n-m} . This message is encrypted by PK_{glm} .

8. $\text{IoTL}_m \rightarrow \text{MME}$: Upon reception of the $M7_{(n-m)}$ message from each IoTD_{n-m} , the IoTL_m checks received AUTH_{n-m} and compares it with generated one. The IoTL_m can authenticate each IoTD_{n-m} by checking whether received $\text{AUTH}_{n-m} =$ calculated AUTH_{n-m} or not. If they match, each IoTD_{n-m} is a valid node. Otherwise, IoTL_m sends AKA failure message to the IoTD_{n-m} . After successful authentication, the group leader computes MXRES using MK, function F2, and RN_{hss} in similar way as HSS computed

it. Then, the IoTL_m employs generated MXRES and received XRES_{n-m} (from all IoTDs) to calculate the XRESL_m . In addition, the IoTL_m produces an Updated Home Network ID (UHNI) for re-authentication phase; $\text{UHNI} = \text{AuthFunc}((\text{RN}_{gm} \oplus \text{RN}_{hss}), \text{AUTH}_m)$. Finally, the IoTL_m forwards $\{\text{XRESL}_m\}_{\text{inv}}(\text{PK}_{glm})$ to the MME via $\text{M8}_{(m)}$ encrypted by PK_{mme} .

9. MME: The MME receives message $\text{M8}_{(m)}$ from the IoTL_m . After retrieving received XRESL_m , it verifies whether saved XRESL_m is same as received XRESL_m or not. If they match, the MME validates the IoT group. Otherwise, the MME declines AKA request and sends an AKA failure message to IoTL_m . The MME also generates the UHNI value the same as IoTL_m to use in re-authentication phase.

7.2.3 Re-Authentication phase

After a successful AKA procedure, an IoT group might send a new attach request to the MME for the n^{th} time. In this case, the involved entities (i.e., IoTDs and MME) use previously shared/generated parameters to perform a lighter AKA process. Therefore, network performance can be improved considerably. There are three kinds of re-authentication in REPS-AKA5 protocol including complete re-authentication, external re-authentication, and internal re-authentication.

7.2.3.1 Complete re-authentication

In this case, an IoT group wants a re-attach to the system completely. The steps of this re-authentication phase are presented as follows (Figure 76):

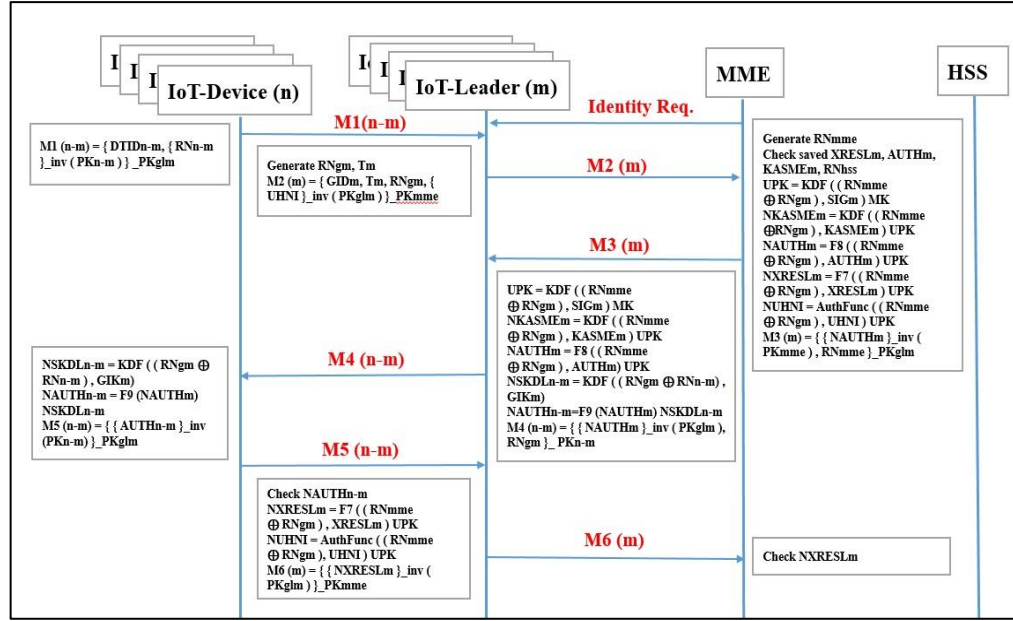


Figure 76: Steps of REPS-AKA5 (Complete re-authentication)

1. $IoT_{D_{n-m}} \rightarrow IoT_{L_m}$: All previously registered IoTDS request authentication from the same MME. Each $IoT_{D_{n-m}}$ generates a new RN_{n-m} , and its own temporary identity (i.e., $DTID_{n-m}$) using its IMSI value. Then, all the $IoT_{D_{n-m}}$ forward the $\{RN_{n-m}\}_{inv(PK_{n-m})}$ and $DTID_{n-m}$ to the IoT_{L_m} via $M1_{(n-m)}$ message encrypted by PK_{glm} .

2. $IoT_{L_m} \rightarrow MME$: After acquiring $M1_{(n-m)}$ messages from all the IoTDS, The IoT_{L_m} checks and updates received information in GL_m . After that, the IoT_{L_m} generates a new RN_{gm} and a valid T_m . Then, it retrieves its UHNI and encrypts it by its inverse public key (PK_{glm}). Finally, the IoT_{L_m} sends security parameters including GID_m , T_m , RN_{gm} , and $\{UHNI\}_{inv(PK_{glm})}$ to the MME via $M2_{(m)}$.

3. $MME \rightarrow IoT_{L_m}$: When the MME receives $M2_{(m)}$ message, it decrypts $M2_{(m)}$ and generates its own random number (RN_{mme}). The MME verifies whether received UHNI is same as saved one or not. If they are equal, it continues AKA procedure. Then, the MME retrieves previous security parameters (i.e., $XRESL_m$, $KASME_m$, and

AUTH_m). It also generates an Updated Key (UPK) using RN_{mme}, and RN_{gm}; UPK=KDF ((RN_{mme}⊕RN_{gm}),SIG_m) MK. The MME needs UPK to generate NKASME_m, NAUTH_m, NXRESL_m, and NUHNI; NKASME_m= KDF ((RN_{mme} ⊕ RN_{gm}), KASME_m) UPK, NAUTH_m =F8 ((RN_{mme}⊕RN_{gm}),AUTH_m) UPK, NXRESL_m= F7((RN_{mme}⊕RN_{gm}), XRESL_m)UPK, NUHNI= AuthFunc((RN_{mme} ⊕ RN_{gm}), UHNI) UPK. Moreover, the MME checks if T_m is expired or valid. If holds, then it runs next step. Otherwise, it sends a request failure message to the IoTL_m. Then, the MME forwards M3_(m) to the IoTL_m. The M3_(m) contains {NAUTH_m}_inv(PK_{mme}), and RN_{mme} which are encrypted by PK_{glm}.

4. IoTL_m → IoTD_{n-m}: After receiving M3_(m) from the MME, the IoTL_m produces security keys/parameters (i.e., UPK, NKASME_m, and NAUTH_m) same as MME. Then, the IoTL_m verifies whether the received NAUTH_m matches with calculated NAUTH_m or not. If they are equal, the IoTL_m authenticates the MME and broadcasts the M4_(n-m) to all IoTDs in m^{th} group. Otherwise, the IoTL_m forwards an AKA failure message to the MME node. Then, the IoTL_m calculates a New SKDL_{n-m} (NSKDL_{n-m}) and a New AUTH_{n-m}(NAUTH_{n-m}); NSKDL_{n-m}=KDF((RN_{gm}⊕RN_{n-m}),GIK_m), NAUTH_{n-m}=F9(NAUTH_m)NSKDL_{n-m}. Finally, the IoTL_m forwards M4_(n-m) including {NAUTH_m}_inv(PK_{glm}), and RN_{gm} to the IoTDs. All IoTDs in each group overhear this broadcasting message.

5. IoTD_{n-m} → IoTL_m: When each IoTD_{n-m} receives M4_(n-m) message from group leader, it extracts included parameters and calculates NSKDL_{n-m} and NAUTH_{n-m}. The IoTD_{n-m} generates these parameters by using KDF, and F9. Then, it forwards {NAUTH_{n-m}}_inv(PK_{n-m}) via M5_(n-m) message to the IoTL_m (encrypted by PK_{glm}).

6. IoTL_m → MME: Upon reception of the M5_(n-m) message from each IoTD_{n-m}, the IoTL_m checks received NAUTH_{n-m} and compares it with generated one for authentication. If they are equal, it verifies the IoTD_{n-m} in the group. Otherwise, a defined AKA decline message is transmitted from IoTL_m to that IoTD_{n-m}. After successful authentication, the group leader computes NXRESL_m, and NUHNI in similar way as MME computes them. Finally, the IoTL_m forwards {NXRESL_m}_{inv}(PK_{glm}) to the MME via M6_(m) encrypted by PK_{mme}.

7. MME: The MME receives message M6_(m) from the IoTL_m and checks the received NXRESL_m. The MME verifies whether saved NXRESL_m equals to the received NXRESL_m or not. If they match, the MME authenticates the IoT group. Otherwise, the MME forwards an AKA decline message to IoTL_m.

7.2.3.2 External re-authentication

When an IoTL_m loses its access to the network for some reasons, it needs to connect to the LTE network and run re-authentication procedure. The steps of External re-authentication are presented as follows (see Figure 77):

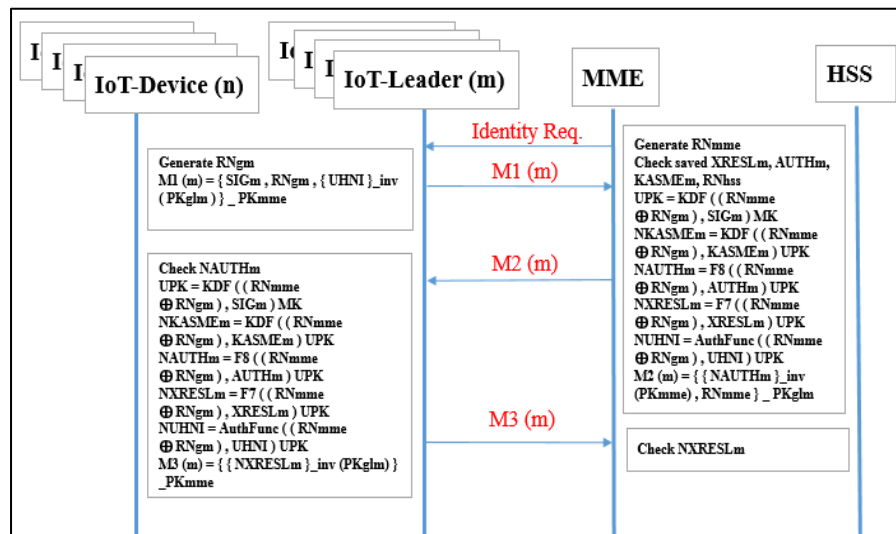


Figure 77: Steps of REPS-AKA5 (External re-authentication)

1. IoTL_m → MME: The IoTL_m generates a new RN_{gm} and retrieves its previous UHNI and SIG_m. Then, IoTL_m sends security parameters including SIG_m, RN_{gm}, and {UHNI}_inv(PK_{glm}) to the MME via M1_(m) message encrypted by PK_{mme}.

2. MME → IoTL_m: After receiving M1_(m) from the IoTL_m, the MME generates its random number (RN_{mme}). Then, the MME retrieves saved SIG_m and compares it with received one. If they are found to be equal, the MME validates the IoTL_m. Otherwise, the MME declines AKA process. Then, the MME retrieves previous security parameters (i.e., XRESL_m, AUTH_m, and KASME_m). It also calculates an Updated Key (UPK); $UPK = KDF((RN_{mme} \oplus RN_{gm}), SIG_m)MK$. The MME needs UPK to generate NKASME_m, NAUTH_m, NXRESL_m, and NUHNI; $NKASME_m = KDF((RN_{mme} \oplus RN_{gm}), KASME_m)UPK$, $NAUTH_m = F8((RN_{mme} \oplus RN_{gm}), AUTH_m)UPK$, $NXRESL_m = F7((RN_{mme} \oplus RN_{gm}), XRESL_m)UPK$, $NUHNI = AuthFunc((RN_{mme} \oplus RN_{gm}), UHNI)UPK$. Then, the MME sends M2_(m) to the IoTL_m. The M2_(m) includes {NAUTH_m}_inv(PK_{mme}), and RN_{mme}.

3. IoTL_m → MME: After receiving M2_(m) from the MME, the IoTL_m acquires NAUTH_m and RN_{mme} from received message. Then, the IoTL_m generates UPK, NKASME_m, NAUTH_m, NXRESL_m, and NUHNI using security functions (i.e., KDF, F7, F8, and AuthFunc). Then, it verifies whether NAUTH_m matches with received one or not. If they match, the IoTL_m authenticates the MME successfully. Then, the IoTL_m forwards {NXRESL_m}_inv(PK_{glm}) to the MME via M3_(m) message. This message is encrypted by PK_{mme}.

4. MME: The MME receives message M3_(m) from the IoTL_m and checks the received NXRESL_m. The MME verifies whether saved NXRESL_m matches with received

$NXRESL_m$ or not. If they are equal, the MME authenticates the IoT group. Otherwise, the MME node sends an AKA failure message to the IoT_L_m .

7.2.3.3 Internal re-authentication

In this case, one or more group member for some reasons lost(s) its/their connection(s) to the group leader. These IoTDs need to re-connect to the IoT group. In this case, the internal re-authentication starts to execute as follows (Figure 78):

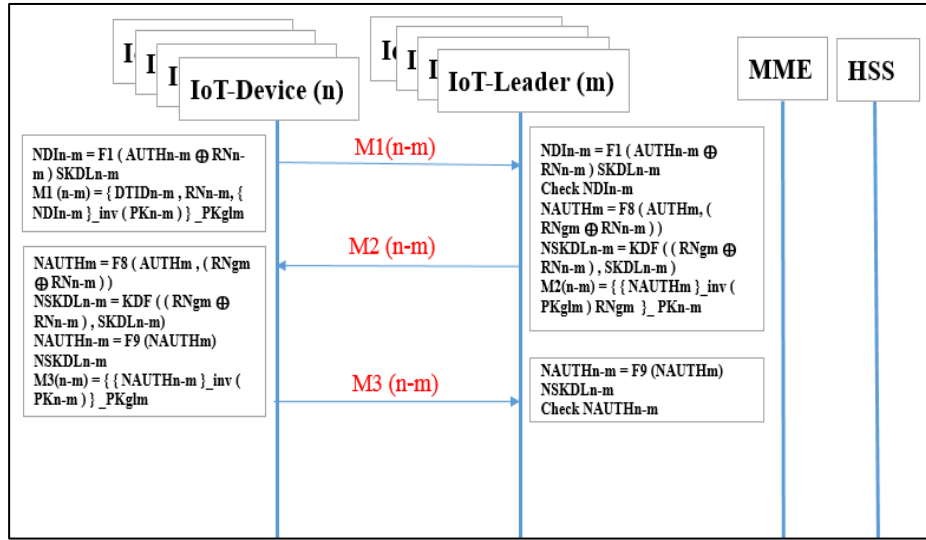


Figure 78: Steps of REPS-AKA5 (Internal re-authentication)

1. $IoT_{n-m} \rightarrow IoT_L_m$: Each previously registered IoT-D forwards a re-attach request to the IoT_L_m . First, the IoT_{n-m} generates a new RN_{n-m} , and calculates a New Device ID (NDI_{n-m}) using its previous parameters (i.e., $AUTH_{n-m}$ and $SKDL_{n-m}$); $NDI_{n-m} = F1(AUTH_{n-m} \oplus RN_{n-m})SKDL_{n-m}$. Then, the IoT_{n-m} forwards $DTID_{n-m}$, RN_{n-m} , and $\{NDI_{n-m}\}_{inv}(PK_{n-m})$ to the IoT_L_m via $M1_{(n-m)}$ message encrypted by PK_{glm} .

2. $IoT_L_m \rightarrow IoT_{n-m}$: Upon receiving $M1_{(n-m)}$ from the IoT_{n-m} , the IoT_L_m retrieves previous security parameters (i.e., $AUTH_{n-m}$, and $SKDL_{n-m}$). The IoT_L_m also produces the NDI_{n-m} using the received RN_{n-m} . Then, it verifies whether the received NDI_{n-m}

matches calculated NDI_{n-m} or not. If they are equal, the $IoTL_m$ authenticates the $IoTD_{n-m}$; otherwise, the $IoTL_m$ forwards an AKA failure message to $IoTD_{n-m}$ and declines the AKA process. Then, the $IoTL_m$ generates a new RN_{gm} , a New $SKDL_{n-m}$ ($NSKDL_{n-m}$), and a New $AUTH_m$ ($NAUTH_m$); $NAUTH_m = F8(AUTH_m, (RN_{gm} \oplus RN_{n-m}))$, $NSKDL_{n-m} = KDF((RN_{gm} \oplus RN_{n-m}), SKDL_{n-m})$. Finally, the $IoTL_m$ forwards $M2_{(n-m)}$ to the $IoTD_{n-m}$. The $M2_{(n-m)}$ includes $\{NAUTH_m\}_{inv}(PK_{glm})$, and RN_{gm} which are encrypted by PK_{n-m} .

3. $IoTD_{n-m} \rightarrow IoTL_m$: When each $IoTD_{n-m}$ receives $M2_{(n-m)}$ message from group leader, it extracts included parameters (i.e., $NAUTH_m$ and RN_{gm}). The $IoTD_{n-m}$ produces $NSKDL_{n-m}$ and $NAUTH_m$ by using security functions (i.e., KDF, and F8). Then, it verifies whether calculated $NAUTH_m$ equals the received $NAUTH_m$ or not. If they are equal, the $IoTD_{n-m}$ validates the $IoTL_m$ successfully. Then, the $IoTD_{n-m}$ computes $NAUTH_{n-m}$; $NAUTH_{n-m} = F9(NAUTH_m)NSKDL_{n-m}$. Then, it forwards $\{NAUTH_{n-m}\}_{inv}(PK_{n-m})$ via $M3_{(n-m)}$ (encrypted by PK_{glm}) to the $IoTL_m$.

4. $IoTL_m$: After receiving the $M3_{(n-m)}$ message from each $IoTD_{n-m}$, the $IoTL_m$ decrypts received message. Then, it generates $NAUTH_{n-m}$ by using $NAUTH_m$ and $NSKDL_{n-m}$ the same as $IoTD_{n-m}$ and compares the generated $NAUTH_{n-m}$ with received one. If they are equal successfully, the $IoTL_m$ validates the $IoTD_{n-m}$. Otherwise, an AKA failure message is sent from $IoTL_m$ to that $IoTD_{n-m}$.

7.2.4 Group updating phase

The group membership of the IoT systems can be updated in some situations. For instance, a new $IoTD$ may join a group or a registered $IoTD$ may leave a group. The joined or removed $IoTD$ may be in different roles (i.e., simple member, group leader,

and backup leader). Group updating can be categorized into two main sections such as group leader/backup leader updating, and group member updating.

7.2.4.1 Group leader/backup leader updating

The group leader update occurs when an IoT leader (i.e., IoTL_m) is exhausted with its defined power sources or when a more appropriate IoT joined the group. If the system requires a leader update because of the first reason, the backup leader is replaced by the current leader. As we mentioned before, there are two leaders in the proposed REPS-AKA5 protocol named “main leader” and “backup leader”. Thus, the backup leader will be main leader and next one in the list (i.e., GLL) will be backup leader and the GLL will be modified accordingly. Thus, a prepared leader list (i.e., GLL) is necessary to specify next leaders quickly since many IoT systems run real-time applications. If the second reason occurs, the leader selection algorithm executes again between the IoTDS. In this case, a newly registered IoT is more appropriate to be a leader than the current leader. The backup leader sends the GLL to the new joined IoT. If new joined IoT finds that it has better features than group leader, it sends a unicast message to backup leader and introduce itself as candidate for leader position. Then, a new leader selection process occurs to find a new leader/backup. Thus, the leader list (i.e., GLL) is updated and then a new (stronger) IoT accepts leader role. During leader updating procedure, backup leaders forward the update messages in the group to identify the best leader based on some predefined properties. These messages are encrypted by GIK_m to provide high level of security level. Moreover, the backup leaders perform this procedure periodically to update the list of the best suggested leaders.

7.2.4.2 Group member updating

According to the REPS-AKA5 update policy, the IoT group needs to cover the forward and backward secrecy in the IoT groups. The goal of forwarding secrecy is that a deleted IoT D has no access to the next communications of the IoT group. Whereas, the backward secrecy goal is that a newly added IoT D is unable to check/retrieve the previous communications in the IoT group. In the proposed method, a secure and dynamic group updating policy is carried out by the IoT Leader. The principal difference between the suggested REPS-AKA5 approach and some previous protocols is that there is no need to run a complete AKA when a new device is added or removed to/from the IoT group. An IoT D may leave the IoT group because of some reasons like: its assigned mission has been done, the IoT D_{n-m} has no more tasks, or it may enter its Save-energy mode to increase its lifetime. When an IoT D is removed, the IoT L_m keeps doing its function till next AKA procedure. Then, the IoT L_m updates GL_m and SIG_m accordingly. Moreover, when the IoT D_{n-m} leaves the IoT group, the IoT L_m revokes all its defined binding relationships with that IoT group and updates the GL_m. Then, the IoT L_m forwards updated GL_m to HSS via MME in the LTE network periodically. The HSS deletes the corresponding rows from the GL_m Table and informs the MME about new updates. The IoT L_m also updates GIK_m frequently to enhance the level of security. Thus, the removed IoT Ds will be unable to have access to the IoT group in the future as a valid registered member. When a new IoT D is added to the system, the IoT leader authenticates the joined device using a Temporary Device ID (TDI_{n-m}) as shown in Figure 79. The steps of the AKA process in group member updating are as follows:

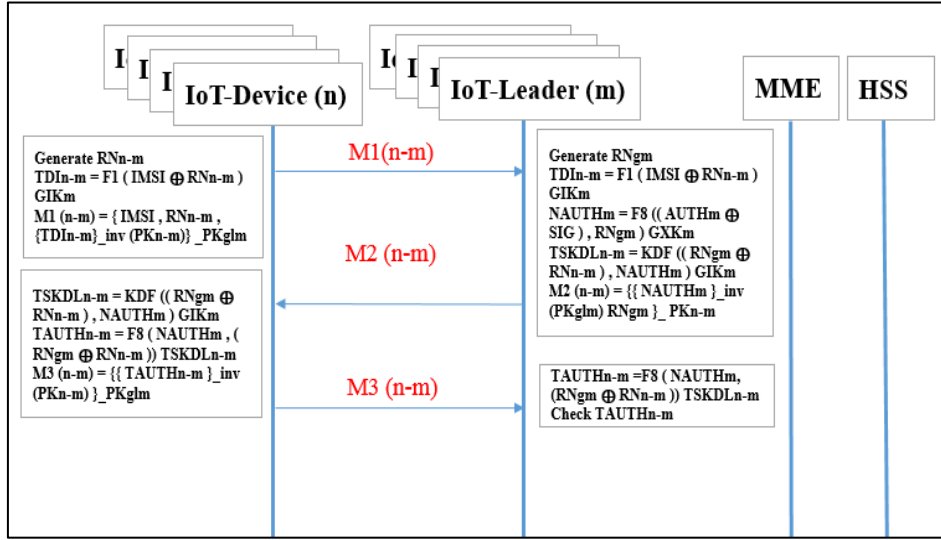


Figure 79: Steps of REPS-AKA5 (Group member updating)

1. $IoT_{D_{n-m}} \rightarrow IoT_{L_m}$: Each new $IoT_{D_{n-m}}$ sends an attach request to the IoT_{L_m} to join the IoT group. First, the $IoT_{D_{n-m}}$ generates a new R_{N-n-m} , and its own TDI_{n-m} using its IMSI and GK_m ; $TDI_{n-m} = F1(IMS_i \oplus R_{N-n-m}) \cdot GK_m$. Then, the $IoT_{D_{n-m}}$ sends IMSI, R_{N-n-m} , and $\{TDI_{n-m}\}_{inv}(PK_{n-m})$ to the IoT_{L_m} via $M1(n-m)$ encrypted by PK_{gim} .

2. $IoT_{L_m} \rightarrow IoT_{D_{n-m}}$: Upon receiving $M1(n-m)$, the IoT_{L_m} retrieves previous security parameters (i.e., $AUTH_{n-m}$, and SIG_m). Then it generates TDI_{n-m} , $NAUTH_m$, and Temporary SKDL_{n-m} ($TSKDL_{n-m}$); $TDI_{n-m} = F1(IMS_i \oplus R_{N-n-m}) \cdot GK_m$, $NAUTH_m = F8((AUTH_m \oplus SIG_m), R_{Ngm}) \cdot GK_m$, $TSKDL_{n-m} = KDF((R_{Ngm} \oplus R_{N-n-m}), NAUTH_m) \cdot GK_m$ where R_{Ngm} is a new random number generated by IoT_{L_m} . Then, the IoT_{L_m} forwards $M2(n-m)$ including $\{NAUTH_m\}_{inv}(PK_{gim})$, R_{Ngm} to the $IoT_{D_{n-m}}$. The $M2(n-m)$ is encrypted by PK_{n-m} .

3. $IoT_{D_{n-m}} \rightarrow IoT_{L_m}$: When new $IoT_{D_{n-m}}$ receives $M2(n-m)$ message, it extracts included parameters and calculates $TSKDL_{n-m}$ the same as IoT_{L_m} using security functions (i.e., KDF). Then, it computes Temporary AUTH_{n-m} ($TAUTH_{n-m}$); $TAUTH_{n-m}$

$m = F8(NAUTH_m, (RN_{gm} \oplus RN_{n-m})TSKDL_{n-m})$. The IoT_{n-m} forwards $\{TAUTH_{n-m}\}_{inv(PK_{n-m})}$ to the $IoTL_m$ via $M3_{(n-m)}$ encrypted by PK_{glm} .

4. IoTL_m: After receiving the $M3_{(n-m)}$ message from each IoT_{n-m} , the $IoTL_m$ decrypts received message. Then, it generates $TAUTH_{n-m}$ by using $NAUTH_m$ and $TSKDL_{n-m}$ the same as IoT_{n-m} . The $IoTL_m$ compares the generated $TAUTH_{n-m}$ with the received one. If they match successfully, the $IoTL_m$ authenticates IoT_{n-m} in group temporarily. Otherwise, an AKA failure message is forwarded from $IoTL_m$ to that IoT_{n-m} . After successful addition of the new IoT_{n-m} into the IoT group, the $IoTL_m$ sends a message to the MME and updates GL_m for next AKA procedure.

7.3 Security analysis of REPS-AKA5 protocol

In section, we provide the security analysis of the REPS-AKA5 method by using BAN Logic, Scyther, and AVISPA tools as follows:

7.3.1 Formal analysis using AVISPA

The AVISPA tool can verify that whether a presented method is safe or not according to the defined goals. The most important goals of the proposed REPS-AKA5 approach are strong mutual authentication and robust key agreement between involved nodes. In the presented method, there are four roles including IoT Device (i.e., IoT_{n-m}), IoT Leader (i.e., $IoTL_m$), MME, and HSS. We verify the REPS-AKA5 method using OFMC and CL-AtSe back-ends modes. (AVISPA codes are presented in Appendix 1). The results of OFMC and CL-AtSe are illustrated in Figure 80 and Figure 81 respectively (Details are in Appendix1).

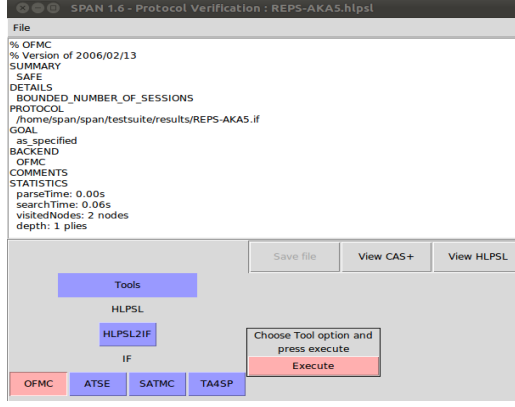


Figure 80: Evaluation result in OFMC

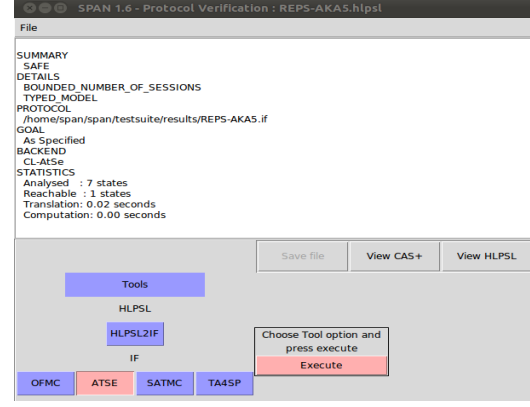


Figure 81: Evaluation result in CL-AtSe

7.3.2 Formal analysis using Scyther

In this section, we evaluate the REPS-AKA5 by means of Scyther tool. The description of the proposed REPS-AKA5 model is implemented according to the defined nodes. There are four nodes in REPS-AKA5. (Scyther codes are presented in Appendix 2).

Based on IoT_D's viewpoint in REPS-AKA5, the security claims are presented as follows: *claim_iotd1* (IOTD, Secret, RN_{nm}); *claim_iotd2* (IOTD, Secret, DTID_{nm}); *claim_iotd3* (IOTD, Secret, XRES_{nm}); *claim_iotd4* (IOTD, Secret, AUTH_{nm}); *claim_iotd5* (IOTD, Secret, NDIn_m); *claim_iotd6* (IOTD, Secret, NAUTH_{nm}); *claim_iotd7* (IOTD, Secret, IMSI); *claim_iotd8* (IOTD, Secret, TDIn_m); *claim_iotd9* (IOTD, Secret, TAUTH_{nm}); *claim_iotda* (IOTD, Alive); *claim_iotdw* (IOTD, Weakagree); *claim_iotdnia* (IOTD, Niagree); *claim_iotdnis* (IOTD, Nisynch);

Based on IoT_L's viewpoint in REPS-AKA5, the security claims are presented as follows: *claim_iotl1* (IOTL, Secret, GID_m); *claim_iotl2* (IOTL, Secret, SIG_m); *claim_iotl3* (IOTL, Secret, T_m); *claim_iotl4* (IOTL, Secret, RN_{gm}); *claim_iotl5* (IOTL, Secret, AUTH_m); *claim_iotl6* (IOTL, Secret, RN_{hss}); *claim_iotl7* (IOTL, Secret, XRES_{lm}); *claim_iotl8* (IOTL, Secret, UHNI); *claim_iotl9* (IOTL, Secret, NAUTH_m); *claim_iotl10* (IOTL, Secret, NXRES_{lm}); *claim_iotla* (IOTL, Alive); *claim_iotlw* (IOTL, Weakagree); *claim_iotlnia* (IOTL, Niagree); *claim_iotlnis* (IOTL, Nisynch);

Based on MME's viewpoint in REPS-AKA5, the security claims are presented as follows:
 $claim_mme1(MME, Secret, GIDm); claim_mme2(MME, Secret, HNID); claim_mme3(MME, Secret, RNgm); claim_mme4(MME, Secret, SIGm); claim_mme5(MME, Secret, AUTHm); claim_mme6(MME, Secret, RNhss); claim_mme7(MME, Secret, NAUTHm); claim_mmea(MME, Alive); claim_mmew(MME, Weakagree); claim_mmenia(MME, Niagree); claim_mmenis(MME, Nisynch);$

Based on HSS's viewpoint in REPS-AKA5, the security claims are presented as follows:
 $claim_hss1(HSS, Secret, XRESIm); claim_hss2(HSS, Secret, AUTHm); claim_hss3(HSS, Secret, KASME m); claim_hss4(HSS, Secret, RNhss); claim_hssa(HSS, Alive); claim_hssw(HSS, Weakagree); claim_hssnia(HSS, Niagree); claim_hssnis(HSS, Nisynch);$

The evaluation results of REPS-AKA5 protocol are presented in Figures 82-85. As it can be seen, the proposed protocols satisfy all defined security claims.

Claim				Status		Comments
REPSAKA5	IOTD	REPSAKA5,iotd1	Secret RNnm	Ok	Verified	No attacks.
		REPSAKA5,iotd2	Secret F1(RNnm,IMSI,GIKm)	Ok	Verified	No attacks.
		REPSAKA5,iotd3	Secret F6(DCIDnm,RNhss,SKnm)	Ok	Verified	No attacks.
		REPSAKA5,iotd4	Secret F9(F8(F1(RNgm,XOR(DCID1,DCIDROG),GXKm),XOR(...	Ok	Verified	No attacks.
		REPSAKA5,iotd5	Secret F1(XOR(F9(F8(F1(RNgm,XOR(DCID1,DCIDROG),GXK...)	Ok	Verified	No attacks.
		REPSAKA5,iotd6	Secret F9(F8(XOR(RNmme,RNgm),F8(F1(RNgm,XOR(DCID1,...	Ok	Verified	No attacks.
		REPSAKA5,iotd7	Secret IMSI	Ok	Verified	No attacks.
		REPSAKA5,iotd8	Secret F1(XOR(IMSI,RNnm),GIKm)	Ok	Verified	No attacks.
		REPSAKA5,iotd9	Secret F8(F8(XOR(RNmme,RNgm),F8(F1(RNgm,XOR(DCID1,...	Ok	Verified	No attacks.
		REPSAKA5,iotda	Alive	Ok	Verified	No attacks.
		REPSAKA5,iotdw	Weakagree	Ok	Verified	No attacks.
		REPSAKA5,iotdnia	Niagree	Ok	Verified	No attacks.
		REPSAKA5,iotdnis	Nisynch	Ok	Verified	No attacks.

Figure 82: Scyther evaluation for defined claims in the IoTD (REPS-AKA5)

Claim				Status	Comments
REPSAKA5	MME	REPSAKA5,mme1	Secret GIDm	Ok	Verified No attacks.
		REPSAKA5,mme2	Secret NIFunc(XOR(MVP,HVP))	Ok	Verified No attacks.
		REPSAKA5,mme3	Secret RNgm	Ok	Verified No attacks.
		REPSAKA5,mme4	Secret F1(RNgm,XOR(DCID1,DCIDROG),GXKm)	Ok	Verified No attacks.
		REPSAKA5,mme5	Secret F8(F1(RNgm,XOR(DCID1,DCIDROG),GXKm),XOR(RNg...	Ok	Verified No attacks.
		REPSAKA5,mme6	Secret RNhss	Ok	Verified No attacks.
		REPSAKA5,mme7	Secret F8(XOR(RNmme,RNgm),F8(F1(RNgm,XOR(DCID1,DCI...	Ok	Verified No attacks.
		REPSAKA5,mme8	Secret RNmme	Ok	Verified No attacks.
		REPSAKA5,mmea	Alive	Ok	Verified No attacks.
		REPSAKA5,mmew	Weakagree	Ok	Verified No attacks.
		REPSAKA5,mmeria	Niagree	Ok	Verified No attacks.
		REPSAKA5,mmeris	Nisynch	Ok	Verified No attacks.

Figure 83: Scyther evaluation for defined claims in the MME (REPS-AKA5)

Claim				Status	Comments
REPSAKA5	IoTl	REPSAKA5,iotl1	Secret GIDm	Ok	Verified No attacks.
		REPSAKA5,iotl2	Secret F1(RNgm,XOR(DCID1,DCIDROG),GXKm)	Ok	Verified No attacks.
		REPSAKA5,iotl3	Secret Tm	Ok	Verified No attacks.
		REPSAKA5,iotl4	Secret RNgm	Ok	Verified No attacks.
		REPSAKA5,iotl5	Secret F8(F1(RNgm,XOR(DCID1,DCIDROG),GXKm),XOR(RNg...	Ok	Verified No attacks.
		REPSAKA5,iotl6	Secret RNhss	Ok	Verified No attacks.
		REPSAKA5,iotl7	Secret F7(F2(XOR(RNhss,RNgm),GXKm),XOR(F6(DCIDnm,R...	Ok	Verified No attacks.
		REPSAKA5,iotl8	Secret AUTHFunc(XOR(RNgm,RNhss),F8(F1(RNgm,XOR(DCI...	Ok	Verified No attacks.
		REPSAKA5,iotl9	Secret F8(XOR(RNmme,RNgm),F8(F1(RNgm,XOR(DCID1,DCI...	Ok	Verified No attacks.
		REPSAKA5,iotl10	Secret F7(XOR(RNmme,RNgm),F7(F2(XOR(RNhss,RNgm),GX...	Ok	Verified No attacks.
		REPSAKA5,iotla	Alive	Ok	Verified No attacks.
		REPSAKA5,iotlw	Weakagree	Ok	Verified No attacks.
		REPSAKA5,iotlnia	Niagree	Ok	Verified No attacks.
		REPSAKA5,iotlnis	Nisynch	Ok	Verified No attacks.

Figure 84: Scyther evaluation for defined claims in the IoTl (REPS-AKA5)

Claim				Status		Comments
REPSAKA5	HSS	REPSAKA5,hss1	Secret F7(F2(XOR(RNhss,RNgm),GXkm),XOR(F6(DCIDnm,R...	Ok	Verified	No attacks.
		REPSAKA5,hss2	Secret F8(F1(RNgm,XOR(DCID1,DCIDROG),GXkm),XOR(RNg...	Ok	Verified	No attacks.
		REPSAKA5,hss3	Secret KDF(RNhss,KDF(XOR(SQN,F5(RNhss,KDF(XOR(RNhs...	Ok	Verified	No attacks.
		REPSAKA5,hss4	Secret RNhss	Ok	Verified	No attacks.
		REPSAKA5,hssa	Alive	Ok	Verified	No attacks.
		REPSAKA5,hssw	Weakagree	Ok	Verified	No attacks.
		REPSAKA5,hssnia	Niagree	Ok	Verified	No attacks.
		REPSAKA5,hssnis	Nisynch	Ok	Verified	No attacks.

Figure 85: Scyther evaluation for defined claims in the HSS (REPS-AKA5)

7.3.3 Security analysis by BAN logic

Security correctness of the proposed REPS-AKA5 are examined by using the BAN Logic. This Logic formally verifies the transmitted messages and checks confidentiality among involved nodes in different phases of REPS-AKA5 method.

The formal messages of the suggested REPS-AKA5 protocol are as follows:

$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m: \{RN_{n-m}, \{DTID_{n-m}\}_{inv}(PK_{n-m})\}_{PK_{glm}}, M2_{(m)}: IoTL_m \rightarrow MME: \{GID_m, \{SIG_m\}_{inv}(PK_{glm}), T_m, RN_{gm}\}_{PK_{mme}}, M3_{(m)}: MME \rightarrow HSS: \{GID_m, \{HNID\}_{inv}(PK_{mme}), RN_{gm}, SIG_m\}_{SK_{mh}}, M4_{(m)}: HSS \rightarrow MME: \{XRESL_m, \{AUTH_m\}_{inv}(PK_{hss}), KASME_m, RN_{hss}\}_{SK_{mh}}, M5_{(m)}: MME \rightarrow IoTL_m: \{\{AUTH_m\}_{inv}(PK_{mme}), RN_{hss}, HNID\}_{PK_{glm}}, M6_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}: \{\{AUTH_m\}_{inv}(PK_{glm}), RN_{gm}, RN_{hss}\}_{PK_{n-m}}, M7_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m: \{\{XRES_{n-m}\}_{inv}(PK_{n-m}), AUTH_{n-m}\}_{PK_{glm}}, M8_{(m)}: IoTL_m \rightarrow MME: \{\{XRESL_m\}_{inv}(PK_{glm})\}_{PK_{mme}}.$

We need to transform all messages into idealized logical format as follows:

$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m: IoTL_m \triangleleft \{\#(RN_{n-m}), \{IoTD_{n-m} \mid \Rightarrow DTID_{n-m}\} PK_{n-m}^{-1}\}_{PK_{glm}}, M2_{(m)}: IoTL_m \rightarrow MME: MME \triangleleft \{\{IoTL_m \mid \Rightarrow SIG_m\}_{PK_{glm}^{-1}}, \#(T_m), \#(RN_{gm})\}_{PK_{mme}}, M3_{(m)}: MME \rightarrow HSS: HSS \triangleleft \{IoTL_m \mid \Rightarrow SIG_m, \{MME \mid \Rightarrow HNID\} PK_{mme}^{-1}, \#(RN_{gm})\}_{SK_{mh}}, M4_{(m)}: HSS \rightarrow MME: MME \triangleleft \{HSS \mid \Rightarrow XRESL_m, \{HSS \mid \Rightarrow AUTH_m\} PK_{hss}^{-1}, HSS \mid \Rightarrow KASME_m, \#(RN_{hss})\}_{SK_{mh}}, M5_{(m)}: MME \rightarrow IoTL_m: IoTL_m \triangleleft \{\{HSS \mid \Rightarrow AUTH_m\}_{PK_{mme}^{-1}}, \#(RN_{hss})\}_{PK_{glm}}, M6_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}: IoTD_{n-m} \triangleleft \{\{HSS \mid \Rightarrow AUTH_m\}_{PK_{glm}^{-1}}, \#(RN_{gm}), \#(RN_{hss})\}_{PK_{n-m}}, M7_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m: IoTL_m \triangleleft \{IoTD_{n-m} \mid \Rightarrow XRES_{n-m}, \{IoTD_{n-m} \mid \Rightarrow AUTH_{n-m}\}\}_{PK_{n-m}}.$

$m | \Rightarrow AUTH_{n-m}, \{IoT_{n-m} | \Rightarrow XRES_{n-m}\} PK_{n-m}^{-1} \} PK_{glm}, \mathbf{M8} \ (m): \mathbf{IoTL}_m \rightarrow \mathbf{MME}: MME \triangleleft \{\{IoT_m | \Rightarrow XRESL_m\} PK_{glm}^{-1} \} PK_{mme}.$

The following initial Assumptions (A) are considered to evaluate the suggested method: **A1**: It is considered that the secret keys (i.e., GK_m , GK_m , SK_{n-m} , and SK_{mh}) and public keys (PK_{mme} , PK_{glm} , PK_{n-m} , and PK_{hss}) are shared between involved entities

as follows: $\mathbf{IoTD}_{n-m} \equiv IoT_{n-m} \xleftrightarrow{GK_m} IoTL_m \wedge IoT_{n-m} \xleftrightarrow{SK_{n-m}} HSS \wedge \xrightarrow{PK_{glm}} IoTL_m, \mathbf{IoTL}_m \equiv IoT_{n-m} \xleftrightarrow{GK_m} IoTL_m \wedge IoTL_m \xleftrightarrow{GK_m} HSS \wedge \xrightarrow{PK_{mme}} MME \xrightarrow{PK_{n-m}} IoT_{n-m}, \mathbf{MME} \equiv MME \xleftrightarrow{SK_{mh}} HSS \wedge \xrightarrow{PK_{glm}} IoTL_m \wedge \xrightarrow{PK_{hss}} HSS, \mathbf{HSS} \equiv MME \xleftrightarrow{SK_{mh}} HSS \wedge \xrightarrow{PK_{mme}} MME, \mathbf{A2}$: It is assumed that the entities calculate

(i.e., control) some security parameters and generated keys as follows: $\mathbf{IoTD}_{n-m} | \Rightarrow DTID_{n-m} \wedge XRES_{n-m} \wedge SKDL_{n-m} \wedge AUTH_{n-m}, \mathbf{IoTL}_m | \Rightarrow SIG_m \wedge MXRES \wedge XRESL_m \wedge UHNI \wedge MK \wedge CK \wedge IK \wedge AK \wedge KASME \wedge KASME_m \wedge AUTH_m \wedge SKDL_{n-m} \wedge AUTH_{n-m}, \mathbf{MME} | \Rightarrow UHNI, \mathbf{HSS} | \Rightarrow SIG_m \wedge MXRES \wedge XRESL_m \wedge XRES_{n-m} \wedge MK \wedge CK \wedge IK \wedge AK \wedge KASME \wedge KASME_m \wedge AUTH_m, \mathbf{A3}$: We consider

that the entities generate some random value to use in key derivation and authentication functions as follows: $\mathbf{IoTD}_{n-m} | \equiv \#(RN_{n-m}), \mathbf{IoTL}_m | \equiv \#(RN_{gm}), \mathbf{MME} | \equiv \#(RN_{mme}), \mathbf{HSS} | \equiv \#(RN_{hss}), \mathbf{A4}$: It is considered that the entities have some security value

related to other involved nodes as follows: $\mathbf{IoTD}_{n-m} \equiv (IoT_{n-m} \xleftrightarrow{IMSI} HSS) \wedge (IoTL_m \xleftrightarrow{DCID_{n-m}} HSS), \mathbf{IoTL}_m \equiv IoTL_m \xleftrightarrow{DCID_{n-m}} HSS, \mathbf{MME} \equiv (MME \xleftrightarrow{HVP} HSS) \wedge (MME \xleftrightarrow{MVP} HSS), \mathbf{HSS} \equiv (MME \xleftrightarrow{HVP} HSS) \wedge (MME \xleftrightarrow{MVP} HSS) \wedge (IoT_{n-m} \xleftrightarrow{IMSI} HSS) \wedge (IoTL_m \xleftrightarrow{DCID_{n-m}} HSS).$

The main security Goals (G) of the suggested protocol are as follows: providing mutual authentication between involved entities (i.e., IoT , $IoTL$, MME , and HSS), and establishing strong session keys between IoT (as user side) and MME (as network side). Detailed security goals are represented as follows: **G1**: The mutual authentication between involved nodes is ensured in the suggested approach as $IoTL_m | \equiv HSS | \equiv MME \wedge MME | \equiv HSS | \equiv IoTL_m$ then $IoTL_m | \equiv MME \wedge MME | \equiv IoTL_m$ and also as $IoTD_{n-m} | \equiv$

$IoTL_m \models HSS \models MME \wedge MME \models HSS \models IoTL_m \models IoTD_{n-m}$ then $IoTD_{n-m} \models MME \wedge MME \models IoTD_{n-m}$

and because $IoTL_m \models HSS \models MME \models IoTD_{n-m} \wedge IoTD_{n-m} \models HSS \models MME \models IoTL_m$ then $IoTL_m \models$

$IoTD_{n-m} \wedge IoTD_{n-m} \models IoTL_m$, **G2:** Key acceptance achieved between different defined

entities. There is a matched KASME between the MME and $IoTL_m$ to present a strong

key agreement. As $MME \models IoTL_m \models KASME_m \wedge IoTL_m \models MME \models KASME_m$ then $IoTL_m \xleftrightarrow{KASME_m}$

MME , **G3:** Message authentication between corresponding entities is satisfied as $IoTD_{n-m}$

$\models IoTL_m \sim AUTH_m$ (in $M6_{(n-m)}$), $IoTL_m \models IoTD_{n-m} \sim DTID_{n-m}$ (in $M1_{(n-m)}$), $IoTL_m \models IoTD_{n-m} \sim$

$XRES_{n-m}$ (in $M7_{(n-m)}$), $IoTL_m \models MME \sim AUTH_m$ (in $M5_{(m)}$), $MME \models IoTL_m \sim SIG_m$ (in $M2_{(m)}$),

$MME \models IoTL_m \sim XRESL_m$ (in $M8_{(m)}$), $MME \models HSS \sim AUTH_m$ (in $M4_{(m)}$), and $HSS \models MME \sim$

$HNID$ (in $M3_{(m)}$), **G4:** Key agreement is satisfied in involved nodes as follows: $IoTD_{n-m}$

$\models IoTL_m \models SKDL_{n-m} \wedge IoTL_m \models IoTD_{n-m} \models SKDL_{n-m}$ then $IoTD_{n-m} \xleftrightarrow{SKDL_{n-m}} IoTL_m$ and also $HSS \models$

$IoTL_m \models MK \wedge IoTL_m \models HSS \models MK$ then $HSS \xleftrightarrow{MK} IoTL_m$, **G5:** The key freshness between

$IoTL_m$ and $IoTD_{n-m}$ in IoT group is provided. The key freshness is also satisfied

between $IoTL_m$ as group side and HSS as network side. The key freshness between

these entities holds since $IoTD_{n-m} \models IoTL_m \models \#(RN_{n-m}) \wedge \#(RN_{gm}) \wedge \#(RN_{hss})$, $IoTL_m \models IoTD_{n-m} \models$

$\#(RN_{n-m}) \wedge \#(RN_{gm}) \wedge \#(RN_{hss})$, $IoTL_m \models HSS \models \#(RN_{gm}) \wedge \#(RN_{hss})$, $HSS \models IoTL_m \models \#(RN_{gm}) \wedge \#(RN_{hss})$

thus, $MK = KDF((RN_{hss} \oplus RN_{gm}), GXX_m)$, $CK = F3(RN_{hss}) MK$, $IK = F4(RN_{hss})$

MK , $AK = F5(RN_{hss}) MK$, $KASME = KDF((SQN \oplus AK), HNID, CK, IK) MK$,

$KASME_m = KDF(RN_{hss}, KASME, SIG_m) GXX_m$, and $SKDL_{n-m} = KDF((RN_{gm} \oplus RN_{n-m}),$

$GIK_m)$ are fresh keys accordingly, **G6:** Confidentiality between the $IoTL_m$ (as a

group side) and the MME (as a network side): $\frac{IoTL_m \models \left(IoTL_m \xleftrightarrow{KASME_m} MME \right), MME \triangleleft \{X\}_{KASME_m}}{IoTL_m \models MME \triangleleft \{X\}_{KASME_m}} \wedge$

$\frac{MME \models \left(IoTL_m \xleftrightarrow{KASME_m} MME \right), IoTL_m \triangleleft \{Y\}_{KASME_m}}{MME \models IoTL_m \triangleleft \{Y\}_{KASME_m}}$ Where ‘X’ and ‘Y’ are transmitted messages

from $IoTL_m$ and MME respectively, **G7:** Confidentiality between the $IoTL_m$ (group

leader) and the $IoTD_{n-m}$ (group

$$\text{device):} \frac{IoTL_m \models (IoTL_m \xrightarrow{SKDL_{n-m}} IoTD_{n-m}), IoTD_{n-m} \triangleleft \{Z\}_{SKDL_{n-m}}}{IoTL_m \models IoTD_{n-m} \sim \{Z\}_{SKDL_{n-m}}} \wedge \frac{IoTD_{n-m} \models (IoTL_m \xrightarrow{SKDL_{n-m}} IoTD_{n-m}), IoTL_m \triangleleft \{R\}_{SKDL_{n-m}}}{IoTD_{n-m} \models IoTL_m \sim \{R\}_{SKDL_{n-m}}}$$

Where ‘Z’ and ‘R’ are transmitted messages from $IoTL_m$ and $IoTD_{n-m}$ respectively.

Then, we evaluate the idealized form of our presented method based on defined hypothesis and the BAN Logic rules. The used BAN Logic rules includes Message meaning rule (R1), Jurisdiction rule (R2), and Nonce Verification rule (R3). We employ the BAN logic rule (R2) to

$$\text{derive:} \frac{HSS \models IoTL_m \Rightarrow SIG_m, HSS \models IoTL_m \models SIG_m}{HSS \models SIG_m}, \frac{IoTL_m \models HSS \Rightarrow AUTH_m, IoTL_m \models HSS \models AUTH_m}{IoTL_m \models AUTH_m},$$

$$\frac{HSS \models MME \Rightarrow HNID, HSS \models MME \models HNID}{HSS \models HNID}, \frac{IoTL_m \models IoTD_{n-m} \Rightarrow AUTH_{n-m}, IoTL_m \models IoTD_{n-m} \models AUTH_{n-m}}{IoTL_m \models AUTH_{n-m}} \text{To satisfy}$$

G1. We utilize the BAN logic rule (R1) to derive:

$$\frac{MME \models (MME \xrightarrow{SK_{mh}} HSS), MME \triangleleft \{KASME_m\}_{SK_{mh}}}{MME \models HSS \sim KASME_m} \wedge IoTL_m \models \Rightarrow KASME_m \text{To satisfy G2. We use the}$$

$$\text{BAN logic rule (R1) to derive:} \frac{IoTD_{n-m} \models (IoTL_m \xrightarrow{PK_{glm}} IoTL_m), IoTD_{n-m} \triangleleft \{AUTH_m\}_{PK_{glm}^{-1}}}{IoTD_{n-m} \models IoTL_m \sim AUTH_m},$$

$$\frac{IoTL_m \models (PK_{n-m} \xrightarrow{IoTD_{n-m}} IoTL_m), IoTL_m \triangleleft \{XRES_{n-m}\}_{PK_{n-m}^{-1}}}{IoTL_m \models IoTD_{n-m} \sim XRES_{n-m}}, \frac{IoTL_m \models (PK_{n-m} \xrightarrow{IoTD_{n-m}} IoTL_m), IoTL_m \triangleleft \{DTID_{n-m}\}_{PK_{n-m}^{-1}}}{IoTL_m \models IoTD_{n-m} \sim DTID_{n-m}},$$

$$\frac{IoTL_m \models (PK_{mme} \xrightarrow{MME} IoTL_m), IoTL_m \triangleleft \{AUTH_m\}_{PK_{mme}^{-1}}}{IoTL_m \models MME \sim AUTH_m}, \frac{MME \models (PK_{glm} \xrightarrow{IoTL_m} MME), MME \triangleleft \{SIG_m\}_{PK_{glm}^{-1}}}{MME \models IoTL_m \sim SIG_m},$$

$$\frac{MME \models (PK_{glm} \xrightarrow{IoTL_m} MME), MME \triangleleft \{XRESL_m\}_{PK_{glm}^{-1}}}{MME \models IoTL_m \sim XRESL_m},$$

$$\frac{MME \models (PK_{hss} \xrightarrow{HSS} MME), MME \triangleleft \{AUTH_m\}_{PK_{hss}^{-1}}}{MME \models HSS \sim AUTH_m}, \frac{HSS \models (PK_{mme} \xrightarrow{MME} HSS), HSS \triangleleft \{HNID\}_{PK_{mme}^{-1}}}{HSS \models MME \sim HNID} \text{To satisfy G3. We}$$

$$\text{employ the BAN logic rule (R2) to derive:} \frac{IoTD_{n-m} \models IoTL_m \Rightarrow SKDL_{n-m}, IoTD_{n-m} \models IoTL_m \models SKDL_{n-m}}{IoTD_{n-m} \models SKDL_{n-m}},$$

$$\frac{IoTL_m \models IoTD_{n-m} \Rightarrow SKDL_{n-m}, IoTL_m \models IoTD_{n-m} \models SKDL_{n-m}}{IoTL_m \models SKDL_{n-m}},$$

$$\frac{HSS \models IoTL_m \Rightarrow MK, HSS \models IoTL_m \models MK}{HSS \models MK}, \frac{IoTL_m \models HSS \Rightarrow MK, IoTL_m \models HSS \models MK}{IoTL_m \models MK} \text{To satisfy G4. Then all derived}$$

keys and parameters from MK and $SKDL_{n-m}$ can be proved accordingly. We employ

the BAN logic rule (R3) to

derive: $\frac{IoTD_{n-m} \models \#(RN_{gm}), IoTD_{n-m} \models IoTL_m \sim RN_{gm}}{IoTD_{n-m} \models IoTL_m \models RN_{gm}} \wedge \frac{IoTD_{n-m} \models IoTL_m \Rightarrow RN_{gm}, IoTD_{n-m} \models IoTL_m \models RN_{gm}}{IoTD_{n-m} \models RN_{gm}}$ Thus,

the MK and all derived keys (i.e., CK, IK, AK, KASME, KASME_m and SKDL_{n-m}) are generated as fresh keys. We applied the same approach to prove key freshness in other entities (i.e., HSS, MME, and IoTL_m) using random values (RN_{hss} and RN_{n-m}). Hence

the protocol satisfies G5. Using proofs in G2 and G4, we found that $IoTL_m \models HSS \models KASME_m$ and $MME \models HSS \models KASME_m$. And by (R2): $\frac{MME \models HSS \Rightarrow KASME_m, MME \models HSS \models KASME_m}{MME \models KASME_m}$

Then, we have $IoTL_m \xleftrightarrow{KASME_m} MME$ to satisfy G6. Using proofs in G4, we found that

$IoTD_{n-m} \models IoTL_m \models SKDL_{n-m} \wedge IoTL_m \models IoTD_{n-m} \models SKDL_{n-m}$. And by G5 and (R3):

$\frac{IoTD_{n-m} \models \#(RN_{gm}), IoTD_{n-m} \models IoTL_m \sim RN_{gm}}{IoTD_{n-m} \models RN_{gm}} \wedge \frac{IoTL_m \models \#(RN_{n-m}), IoTL_m \models IoTD_{n-m} \sim RN_{n-m}}{IoTL_m \models RN_{n-m}}$ we found that

SKDL_{n-m} is fresh and depend on RN_{gm} and RN_{n-m} which are shared between entities.

Thus, we have $IoTL_m \xleftrightarrow{KASME_m} MME$ to satisfy G7. It is worth to mention that we have followed the same approach by using BAN logic rules to provide the security prove for re-authentication phases.

7.3.4 Proof of security requirements

We provide a security analysis of the REPS-AKA5 protocol to demonstrate that the proposed approach can cover the security goals in LTE-enabled IoT systems. Moreover, suggested protocol has been compared with some other security methods with respect to several security goals at the end of this section [69] [42] [10]. In this security analysis, we assumed that there is no private interfaces between involved entities (i.e., IoTD, IoTL, MME and HSS). Thus, an attacker can access to the wireless channels between each IoT device and the IoTL. S/he can also monitor communication channels between each IoTL and network side. We illustrate some correctness proofs for the proposed method with their achievements as follows:

Claim 1: The REPS-AKA5 provides strong message authentication.

Proof: In REPS-AKA5, the transmitted messages must be verified by receiver to ensure that the messages are made by a valid node. The REPS-AKA5 provides a strong message authentication between involved roles properly. The $IoTL_m$ will be sure that the received message is from a valid $IoTD_{n-m}$ because the message $M1_{(n-m)}$ is signed by using inverse public key of the $IoTD_{n-m}$. In the message $M1_{(n-m)}$, the $IoTL_m$ authenticates $IoTD_{n-m}$ using $\{DTID_{n-m}\}_{inv} (PK_{n-m})$. In addition, the involved entity cannot deny its signature in the transmitted messages which are sent to another entity. Message authentication is satisfied in other AKA messages as illustrated in Table 33.

Table 33: Message authentication in REPS-AKA5

Phase	Messages	Authentication Descriptions
First Authentication	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The $DTID_{n-m}$ is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M2_{(m)}: IoTL_m \rightarrow MME$	The SIG_m is authenticated by MME as built by $IoTL_m$
	$M3_{(m)}: MME \rightarrow HSS$	The $HNID$ is authenticated by HSS as built by MME
	$M4_{(m)}: HSS \rightarrow MME$	The $AUTH_m$ is authenticated by MME as built by HSS
	$M5_{(m)}: MME \rightarrow IoTL_m$	The $AUTH_m$ is authenticated by $IoTL_m$ as built by MME
	$M6_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	The $AUTH_m$ is authenticated by $IoTD_{n-m}$ as built by $IoTL_m$
	$M7_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The $XRES_{n-m}$ is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M8_{(m)}: IoTL_m \rightarrow MME$	The $XRESL_m$ is authenticated by MME as built by $IoTL_m$
Complete re-Auth.	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The RN_{n-m} is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M2_{(m)}: IoTL_m \rightarrow MME$	The $UHNI$ is authenticated by MME as built by $IoTL_m$
	$M3_{(m)}: MME \rightarrow IoTL_m$	The $NAUTH_m$ is authenticated by $IoTL_m$ as built by MME
	$M4_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	The $NAUTH_m$ is authenticated by $IoTD_{n-m}$ as built by $IoTL_m$
	$M5_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The $AUTH_{n-m}$ is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M6_{(m)}: IoTL_m \rightarrow MME$	The $NXRESL_m$ is authenticated by MME as built by $IoTL_m$
External re-Auth.	$M1_{(m)}: IoTL_m \rightarrow MME$	The $UHNI$ is authenticated by MME as built by $IoTL_m$
	$M2_{(m)}: MME \rightarrow IoTL_m$	The $NAUTH_m$ is authenticated by $IoTL_m$ as built by MME
	$M3_{(m)}: IoTL_m \rightarrow MME$	The $NXRESL_m$ is authenticated by MME as built by $IoTL_m$
Internal re-Auth.	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The NDI_{n-m} is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M2_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	The $NAUTH_m$ is authenticated by $IoTD_{n-m}$ as built by $IoTL_m$
	$M3_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The $NAUTH_{n-m}$ is verified by $IoTL_m$ as built by $IoTD_{n-m}$
Group member updating	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The TDI_{n-m} is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$
	$M2_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	The $NAUTH_m$ is authenticated by $IoTD_{n-m}$ as built by $IoTL_m$
	$M3_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	The $TAUTH_{n-m}$ is authenticated by $IoTL_m$ as built by $IoTD_{n-m}$

Claim 2: The protocol ensures robust mutual authentication.

Proof: The proposed method need to guarantee that all the involved nodes can provide a strong mutual authentication efficiently. A robust mutual authentication also helps to avoid performance bottlenecks. The REPS-AKA5 approach provides robust authentication between IoTL_m and IoTD_{n-m} , between MME and IoTL_m , and between HSS and MME (i.e., EPC entities). For a robust mutual authentication, the HSS as network side generates AUTH_m and XRESL_m parameters using random numbers and the secret keys. On the IoT group side, IoTL_m also generates these parameters by the same random numbers and received XRES_{n-m} from all the IoTD_{n-m} . The IoTL_m checks received AUTH_m and compares it with generated one in IoT group. If both values are equal, it shows that the HSS is valid node in the home network. It also shows that MME is a genuine node in the serving network authenticated by the HSS. In addition, the MME can verify authenticity of the IoT group (i.e., IoTL_m and all IoTD_{n-m}) by checking equality of the two XRESL_m . These XRESL_m values are sent by HSS and IoTL_m . If they match, it means the IoT group (i.e., all the IoTDs) are valid nodes in LTE-enabled IoT system. We can prove mutual authentication between all other nodes by the same way. The IoTD_{n-m} verifies the IoTL_m by using SKDL_{n-m} , NSKDL_{n-m} , and TSKDL_{n-m} . The IoTL_m also can authenticate each IoTD_{n-m} using AUTH_m , NAUTH_m , AUTH_{n-m} , NDI_{n-m} , TDI_{n-m} , XRES_{n-m} and TAUTH_{n-m} . The group side (i.e., IoTL_m) verifies the network side (i.e., HSS and MME) as by using NAUTH_m , and AUTH_m . The HSS authenticates the IoT group (i.e., IoTL_m and IoTD_{n-m}) by using AUTH_m , SIG_m , SK_{n-m} , and GXX_m . Additionally, the MME verifies the IoTL_m and the IoT group employing XRESL_m , NXRESL_m , and UNHI . In the EPC section, the HSS and MME can authenticate each other using HNID value generated by HVP and MVP.

Consequently, proper mutual authentication is achieved when the involved nodes verify each other. In the REPS-AKA5, all these mentioned parameters are generated using strong calculations, random numbers, and pre-shared secret keys. These secret keys are only known to the valid MME and HSS as LTE network side and the IoTDS. An attacker cannot compromise these secret keys since we never send these pre-shared keys over insecure communication channels in the LTE-enabled IoT system.

Claim 3: The protocol ensures protection of IoTDS's Identity.

Proof: The REPS-AKA5 protocol protects IoTDS's identities in the LTE-enabled IoT systems. In the standard EPS-AKA, each IoTDS_{n-m} has to forward its IMSI value (i.e., user identity) to the MME in clear text. Therefore, an adversary can obtain the transmitted IMSI and launch different malicious attacks. In the suggested method, each IoTDS_{n-m} and the group leader use dynamic IDs, instead of the original IDs. Thus, an attacker is not able to extract the original ID of each IoTDS. Consequently, the privacy requirement are provided appropriately during the REPS-AKA5 procedures. For strong privacy preservation in REPS-AKA5 method, each IoTDS_{n-m} also calculates DTID_{n-m}, NDI_{n-m}, and TDI_{n-m} in different situations (i.e., First Authentication, Internal re-authentication, and Group member updates). Then, the IoTDS_{n-m} encrypts them by using the public keys before transmission. Hence, an attacker can never succeed in extracting the original identity of each IoTDS. In addition, the IoTL_m aggregates the DCID_{n-m} of all IoTDS_{n-m} into one SIG_m and then forwards it to the MME. Moreover, the MME uses secret key (i.e., SK_{mh}) to encrypt the received GID_m, and SIG_m as group identifiers before sending them to the HSS in the home network. In group member updating, the REPS-AKA5 also provides strong privacy by employing dynamic device identities. For each authentication process, a unique and dynamic device ID is made

by random value (i.e., RN_{n-m}) and proper secret key (i.e., GK_m) whenever a new IoT_{n-m} connects to the IoT group. Thus, the suggested method provides the privacy of each IoTD considerably.

Claim 4: The protocol ensures resistance to the MitM attack.

Proof: The secure session keys and dynamic identity of IoTD's protect the LTE-enabled IoT system from the MitM attacks. The proposed REPS-AKA5 protocol ensures secure communication between the involved nodes. Therefore, an attacker cannot forge original ID of IoTDs. Hence, the suggested method is resistant to the MitM attacks properly. A malicious attacker cannot extract the IoTD's identity (i.e., IMSI) since their dynamic and temporary IDs are generated and transmitted instead of original IDs. In addition, other AKA parameters such as SIG_m , $AUTH_m$, $NAUTH_m$, $TAUTH_{n-m}$, and $AUTH_{n-m}$ are not sent in clear text. Hence, in the REPS-AKA5 protocol, the group's identity also cannot be retrieved/modified by an adversary. Additionally, the $NKASME_m$ and $KASME_m$ are derived from generated master key (i.e., MK). The MK is calculated by using random value and GK_m which is shared between IoT group leader and network nodes. As all session keys are derived from these keys (i.e., $NKASME_m$ and $KASME_m$), an attacker cannot retrieve the session keys by the public information. In the REPS-AKA5, the IoT_{n-m} and network nodes (i.e., HSS and MME) locally calculate these master keys and derived session keys with pre-shared secret keys (i.e., GK_m). The GK_m key also is not forwarded over insecure communication channels. Thus, the attackers are not be able to generate the session keys, even if they steal some AKA parameters during transmissions. Moreover, all AKA messages are encrypted and embedded with a random number and timestamp. These timestamps prevent the AKA transmitted messages from being reused by

attackers. Consequently, MitM attacks can be thwarted in LTE-enabled IoT systems by using pre-shared keys, random values, and dynamic/temporary IDs.

Claim 5: The REPS-AKA5 can optimize generation overload of AKA parameters.

Proof: In REPS-AKA5, the $NKASME_m$ are generated by using shared random numbers, updated key (i.e., UPK) and the $KASME_m$. Hence, in re-authentication phases, there is no need to calculate $KASME_m$ in group/network side. This manner helps to decrease key generation overload. Furthermore, the AKA parameters such as $NAUTH_m$, $NUHNI$, $NXRESL_m$, $NAUTH_{n-m}$, and $NSKDL_{n-m}$ are generated by random numbers and UPK to optimize the overload of AKA parameter generation.

Claim 6: The protocol ensures resistance to SNID leakage.

Proof: The HSS in home network can authenticate the MME in serving network by generating and verifying HNID value. The HNID is calculated by using the MVP and HVP values which are pre-shared and stored in the HSS and MME. The HNID is encrypted by inverse public key and SK_{mh} , which means that the adversary has to know the keys to encrypt the HNID. The EPC nodes update these parameters frequently according to the system policies. Thus, by using REPS-AKA5, the HSS protects the identity of MME from being impersonated even if its SNID parameter is extracted.

Claim 7: The REPS-AKA5 ensures resistance to single-key dependency.

Proof: One of the most significant features of the presented protocol is that it overcomes the single key issue of the standard EPS-AKA protocol. The REPS-AKA5

ensures resistance to the single-key issue properly. In the EPS-AKA, the UE (i.e., IoT_D) and the HSS employ a symmetric key to run AKA protocol. If this key is obtained by an attacker, s/he can extract all the subsequent keys. In this case, the secrecy of whole system will be at risk. To avoid this issue, the proposed method uses some fresh and dynamic keys instead of the permanent one. These keys are reallocated after each successful AKA procedure in LTE-enabled IoT system. In the REPS-AKA5 protocol, these generated keys are linked with original keys (i.e. GXK_m), SIG_m , and random numbers. For each new authentication, the IoT_L_m and network nodes employ a new random number to produce fresh master keys accordingly. The main goal of the MK and UPK is to protect LTE-enabled IoT system from single key issues. These master keys (i.e., MK and UPK) and derived parameters (i.e., $XRES_{n-m}$, $XRES_{L_m}$ and $AUTH_m$) are created by random numbers and one-way hash functions. This way make them very difficult to be extracted. Even if an attacker compromises the pre-shared keys, s/he can never obtain MK and UPK. Therefore, s/he cannot extract other session keys and security parameters. The MK and UPK are used to generate fresh $KASME_m$ and $NKASME_m$ which are main keys to calculate other keys in REPS-AKA5. Thus, the presented method avoids the single key issues in AKA procedure appropriately.

Claim 8: The REPS-AKA5 can provide confidentiality in control plan.

Proof: In the EPS-AKA protocol, adversaries can steal transmitted messages in control plan. But, the REPS-AKA5 protocol ensures resistance to eavesdropping attacks by using a combination of asymmetric and symmetric algorithms. These algorithms are used between involved nodes including IoT_D_{n-m}, IoT_L_m, MME, and HSS. By using the suggested approach, attackers are unable to eavesdrop on security parameters transmitted through wire/wireless links. Therefore, the protocol is safe under this

adversary model. We use a combination of symmetric and asymmetric algorithms to provide proper confidentiality for all the wire/wireless interfaces. Hence, this approach is also resistance to DoS and MitM attacks. These attacks are feasible because of sending unprotected message to the network nodes. In the REPS-AKA5, all transmitted parameters are encrypted by a secret keys and public keys, as shown in Table 34 (Asym: Asymmetric, Sym: Symmetric). Therefore, confidentiality requirements can be satisfied by the REPS-AKA5 protocol completely. When the proposed AKA method is completed, the IoT group and the MME agree upon $KASME_m/NKASME_m$. Then, these nodes derive related NAS and AS keys to ensure confidentiality and integrity protection in the NAS/AS layers.

Table 34: Asymmetric and symmetric keys in REPS-AKA5

Phase	Messages	Encryption types	keys
First Authentication	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M2_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
	$M3_{(m)}: MME \rightarrow HSS$	Sym	SK_{mh}
	$M4_{(m)}: HSS \rightarrow MME$	Sym	SK_{mh}
	$M5_{(m)}: MME \rightarrow IoTL_m$	Asym	PK_{glm}
	$M6_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	Asym	PK_{n-m}
	$M7_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M8_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
Complete re-Auth.	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M2_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
	$M3_{(m)}: MME \rightarrow IoTL_m$	Asym	PK_{glm}
	$M4_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	Asym	PK_{n-m}
	$M5_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M6_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
External re-Auth.	$M1_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
	$M2_{(m)}: MME \rightarrow IoTL_m$	Asym	PK_{glm}
	$M3_{(m)}: IoTL_m \rightarrow MME$	Asym	PK_{mme}
Internal re-Auth.	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M2_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	Asym	PK_{n-m}
	$M3_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
Group member updating	$M1_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}
	$M2_{(n-m)}: IoTL_m \rightarrow IoTD_{n-m}$	Asym	PK_{n-m}
	$M3_{(n-m)}: IoTD_{n-m} \rightarrow IoTL_m$	Asym	PK_{glm}

Claim 9: The REPS-AKA5 ensures prevention of replay attacks.

Proof: The REPS-AKA5 method provides a perfect resistance to the replay attacks. In our method, the involved nodes (i.e., IoTD_{n-m} , IoTL_m , MME, and HSS) employ some fresh random number (i.e., RN_{n-m} , RN_{gm} , RN_{mme} , and RN_{hss}) and fresh timestamp (i.e., T_m) to withstand the replay attacks. Therefore, when an adversary wants to resent a previously used request message, s/he cannot generate a valid message by reusing previous random numbers. The MME verifies the request message (i.e., $M2_{(m)}$) using the T_m value which marks the release time of each transmitted message. If the T_m is a valid timestamp, then the MME accepts the AKA request from the IoT group. Otherwise, it ignores the received request and does not forward the request to the home network (i.e., HSS node). Thus, random values in transmitted messages can defend against replay attacks in LTE-enabled IoT systems.

Claim 10: The protocol ensures resistance to redirection attack.

Proof: The proposed protocol can be resistant to redirection attacks since it provides a strong mutual authentication by employing a proper challenge-response procedure. In the standard EPS-AKA, an attacker can launch a redirection attack by simulating a fake network node and extracting security information of an IoTD. Then, s/he impersonates him/herself as a valid IoTD and forwards the modified messages to LTE-enabled IoT system. This adversary model fails if that attacker fails to extract IoTD's security parameters by simulating the network nodes. Without these information, the attacker is not able to impersonate an IoTD and connect to a legitimate LTE-enabled IoT system. In the REPS-AKA5, the group leader (i.e., IoTL_m) produces the SIG_m by using security information of all the IoTDs and sends the SIG_m to the LTE network.

Then, the HSS generates and verifies this SIG_m to authenticate IoT group in LTE-enabled IoT system. The AKA request message is ignored if the HSS fails to match the SIG_m forwarded by the MME and the generated one. Hence, the attacker cannot impersonate any IoT group and connect to LTE system. In addition, an identity of each IoT is preserved by RN_{n-m} , and GK_m . Hence, the attacker cannot obtain the original identity of the IoTs and fails to impersonate them. Therefore, it is impossible for a malicious adversary to launch a redirection attack on the LTE-enabled IoT system.

Claim 11: The protocol provides Perfect Forward/Backward Secrecy (PFS/PBS)

Proof: When an AKA protocol provides a PFS and a PBS, even if the secret key of each IoT is compromised by attacker, s/he cannot learn all the following security keys. In the REPS-AKA5 method, the long-term secret keys (i.e., GK_m , GK_m , and SK_{n-m}) are stored in the safe environment of the involved nodes. It is very difficult to be obtained by an adversary. Even if these secret keys are compromised, the following security parameters and generated keys cannot be achieved. Because the master keys (i.e., MK and UPK) are calculated by the random number (i.e., RN_{gm} , RN_{mme} , and RN_{hss}), the suggested approach can ensure the freshness of the derived keys/parameters (i.e., $KASME_m$, $NKASME_m$, $XRESL_m$, and $AUTH_m$). Moreover, in group updating phase, the IoT_{L_m} removes all the previous binding information of the removed IoT. In addition, when a new IoT joins the group, IoT_{L_m} employs new/temporary security parameters (i.e., NDI_{n-m} and TDI_{n-m}) to the newly joined IoT. Thus, on group updating phase, the PBS/PFS can be provided properly. When a new node (i.e., IoT) wants to join the IoT group and starts its AKA process, it does not have access to the security parameters before it is validated by that group leader. Similarly, when an IoT leaves the IoT group for any reason, it fails to use the group communication. Since the

internal key of the group is updated as soon as one IoT-D leaves the IoT group. Accordingly, an attacker is not able to reach to the current group key and derived key (i.e., $NSKDL_{n-m}$) using previous/next keys. Thus, the REPS-AKA5 method can provide the key unlink-ability and PFS/PBS.

Claim 12: The protocol ensures resistance to DoS attack.

Proof: The DoS attacks can be performed by a set of fake IoT-Ds during the AKA procedure. The REPS-AKA5 approach provides a practical congestion avoidance to defend DoS attack by using a group-based approach. The HSS recognizes the modified/forged request messages by checking SIG_m value which is sent by $IoT-L_m$ (i.e., group leader). The HSS generates the SIG_m and compares it with received one from $IoT-L_m$ through the MME. If the HSS finds a mismatch between these two SIG_m parameters, the HSS ignores this access request. As this request is received from an invalid IoT-D in the IoT group, the HSS forwards an AKA decline message to the IoT group through the MME. Thus, an attacker cannot run out the HSS resources and launch a DoS attack easily. On the other hand, each $IoT-L_m$ computes the $NAUTH_m$ and $AUTH_m$ to verify the authenticity of network by comparing them with received ones. If this match fails, the $IoT-L_m$ transmit an AKA failure message to the network nodes. Thus, the presented method can resist the DoS attack in the AKA procedure.

In Tables 35 and 36, we compare the security properties of the proposed protocol with other AKA methods [42] [10] (S: Satisfied; U: Unsatisfied; Asym: Asymmetric; Sym: Symmetric). As it can be seen from the tables, the suggested protocol can resist more security threats/attacks than other AKA methods.

Table 35: Comparison of security properties in Group-based AKA protocols

<i>Security Goals</i>	<i>SE-AKA [58]</i>	<i>EG-AKA [52]</i>	<i>NOVEL-AKA [59]</i>	<i>GROUP-AKA [54]</i>	<i>GBAAM-AKA [53]</i>	<i>PRIVACY-AKA [56]</i>	<i>GLARM-AKA [71]</i>	<i>GR-AKA [60]</i>	<i>GBS-AKA [57]</i>	<i>SEGB-AKA [42]</i>	<i>REPS-AKA5</i>
<i>Type of cryptosystem</i>	Hybrid	Hybrid	Sym	Sym	Asym	Hybrid	Sym	Asym	Sym	Sym	Hybrid
<i>Basic security requirement</i>	S	S	S	S	S	S	S	S	S	S	S
<i>Adaptation to the 3GPP</i>	U	U	S	U	U	U	S	U	S	S	S
<i>Protection of user's privacy</i>	S	S	S	U	U	S	U	S	U	S	S
<i>Signaling congestion avoidance</i>	U	U	U	S	S	S	S	S	S	S	S
<i>provide PFS/PBS features</i>	S	U	S	S	S	U	S	S	U	S	S
<i>Maintain unlink ability of group keys</i>	U	U	U	S	U	U	U	S	U	S	S
<i>Overcome single key issues</i>	U	U	U	U	U	U	U	U	U	U	S
<i>Resistance to redirection attack</i>	S	U	U	S	U	S	S	S	S	S	S
<i>Protection against MitM attacks</i>	S	S	S	S	S	S	S	S	S	S	S
<i>Resistance to impersonation attacks</i>	S	S	S	S	S	S	U	S	U	S	S
<i>Secure against Replay attacks</i>	S	S	S	S	S	S	S	S	S	S	S
<i>Resistance to DoS attacks</i>	S	U	U	U	S	S	S	S	U	U	S
<i>Total number</i>	8	5	7	8	7	8	8	10	6	10	12

Table 36: Security comparison of the Group-based AKA protocols

<i>Security properties</i>	<i>Ref. [61]</i>	<i>Ref. [62]</i>	<i>Ref. [63]</i>	<i>Ref. [74]</i>	<i>Ref. [65]</i>	<i>Ref. [103]</i>	<i>Ref. [24]</i>	<i>Ref. [29]</i>	<i>REPS-AKA5</i>
<i>Cryptography system</i>	Sym	Sym	Asym	Sym	Sym	Hybrid	Sym	Sym	Hybrid
<i>Based on the 3GPP architecture</i>	S	S	U	S	S	S	S	S	S
<i>Ensures privacy protection</i>	U	U	S	U	S	S	S	S	S
<i>Avoid signaling congestion issue</i>	U	U	U	S	U	U	S	S	S
<i>Protection from redirection attack</i>	U	U	U	U	U	S	S	S	S
<i>Resistance to MitM attack</i>	U	U	U	U	U	S	S	S	S
<i>Resistance to replay attack</i>	S	S	S	S	S	S	S	S	S
<i>Protection from impersonation attack</i>	U	U	U	U	U	S	S	S	S
<i>Resistance to DoS attack</i>	U	U	U	U	U	S	S	S	S
<i>PFS/PBS</i>	U	U	U	U	U	U	S	S	S
<i>Avoids the single key issues</i>	U	U	U	U	U	U	S	U	S
<i>Supports to the IoT-based application</i>	U	U	U	U	U	U	U	S	S

7.4 Performance analysis of the proposed model: REPS-AKA5

The performance analysis of the REPS-AKA5 approach is presented with respect to five metrics including total bandwidth consumption, bandwidth consumption of core traffic, computation cost, signaling overhead, and storage overhead. We compare our

protocol with some other suggested methods to illustrate that the REPS-AKA5 brings a significant improvement in terms of the mentioned metrics.

7.4.1 Total bandwidth consumption

To evaluate the Total Bandwidth Consumption (*TBC*) of AKA protocols, we consider that there are “ n ” IoT devices and “ m ” IoT groups, where “ n ” is greater than “ m ” (logically).

The *TBC* of each AKA method can be evaluated in terms of the length (in bits) of the

transmitted messages by (7.1) [12, 24, 41]: $TBC_{AKA} = \sum_{k=1}^{k=i} (MSG)_k$ (7.1)

Where “ i ” is the total number of transmitted messages in AKA method, and $(MSG)_k$ shows the length of the k^{th} message (in bits).

The *TBC* of REPS-AKA5 (First Authentication phase): The REPS-AKA5 uses messages in first authentication phase. The list of the involved parameters and their sizes (in bit) are shown in Table 37. The *TBC* of the REPS-AKA5 protocol can be calculated as: $TBC_{REPS-AKA5} = \sum_{k=1}^{k=8} (MSG)_k = 800n + 1376m$.

Table 37: Size of the used parameters in the REPS-AKA5

Parameter	Size (bit)	Parameter	Size (bit)	Parameter	Size (bit)
$RN_{n-m}/RN_{gm}/RN_{mme}/RN_{hss}$	128	T_m	32	$HNID$	32
$XRES_{n-m}$	32	$IMSI$	128	$(N)UHNI$	32
$(N)XRESL_m$	32	SIG_m	64	TDI_{n-m}	128
$DCID_{n-m}/DTID_{n-m}$	128	GID_m	32	NDI_{n-m}	128
$(N)KASME_m$	256	$(N)AUTH_m$	128	$(N/T)AUTH_{n-m}$	128

The *TBC* of REPS-AKA5 (Complete re-authentication phase): In complete re-authentication phase of the REPS-AKA5, we perform AKA procedure only with 6 messages. The *TBC* of the REPS-AKA5 in this phase of protocol (REPS-AKA5_(C-re-Auth-Ph)) can be presented as: $TBC_{REPS-AKA5(C-re-Auth-Ph)} = \sum_{k=1}^{k=6} (MSG)_k = 640n + 512m$.

The *TBC* of REPS-AKA5 (External re-authentication phase): In External re-authentication, we need 3 messages. The *TBC* of REPS-AKA5 (REPS-AKA5_(E-re-Auth-Ph)) can be presented as: $TBC_{REPS-AKA5(E-re-Auth-Ph)} = \sum_{k=1}^{k=3}(MSG)_k = 512m$.

The *TBC* of REPS-AKA5 (Internal re-authentication phase): In Internal re-authentication, we use 3 messages. The *TBC* of the REPS-AKA5 (REPS-AKA5_(I-re-Auth-Ph)) can be calculated as: $TBC_{REPS-AKA5(I-re-Auth-Ph)} = \sum_{k=1}^{k=3}(MSG)_k = 768n$.

The *TBC* of the AKA protocols per message is presented in Table 38 [42]. We compare the *TBC* of the proposed REPS-AKA5 methods with other methods. The *TBC* ratios of the REPS-AKA5 methods with other protocols for the different values of “*n*” (when *m*=50) are presented in Table 39.

Table 38: The *TBC* (in Bits) of group based AKA protocols (per message)

Messages	SE-AKA [58]	EG-AKA [52]	NOVEL-AKA [59]	GROUP-AKA [54]	GBAAM-AKA [53]	PRIVACY-AKA [56]	GLARM-AKA [71]	GR-AKA [60]	GBS-AKA[57]	SEGB-AKA [42]	REPS-AKA5
<i>M1</i>	512m	576m	544m	128n+128	128n+128	448n	448n	320m	512n	384n	256n
<i>M2</i>	552m	704m	544m	128n+128+128m	128n+128+128m	384n+64m	384n+64m	360m	448n+64m	320n+128m	256m
<i>M3</i>	432m	768m	816m	1200m	128n+736m	384n+192m	384n+104m	256m	448n+64m	320n+168m	256m
<i>M4</i>	1072m	512m	496m	624m	419n	992m	1072m	448m	432m	256n+688m	544m
<i>M5</i>	256m	256m	64m	624n	64n	864m	688m	192n+192	512m	128n+432m	288m
<i>M6</i>	384n	256m	480n	64m	128n	864n	688n	320n	512n	128n+432m	384n
<i>M7</i>	1072n	128m	496n	752n	960n	64n	64n	384n	-	64n	160n
<i>M8</i>	256n	576n	64n	448n	384n+576	64m	64m	-	-	64m	32m
<i>M9</i>	-	512n	-	384m	1024m	-	-	-	-	-	-
<i>M10</i>	-	256n	-	-	1024n	-	-	-	-	-	-
<i>M11</i>	-	256n	-	-	-	-	-	-	-	-	-
<i>M12</i>	-	128n	-	-	-	-	-	-	-	-	-
<i>Total (bit)</i>	1712(n-m)+2824m	1728(n-m)+3200m	1040(n-m)+2464m	2080n+256+2400m	3363n+1888m+832	2144n+2176m	1968n+1992m	896n+192+1384m	1920n+1072m	1600n+1912m	800n+1376m

Table 39: The ratio of *TBC* for different AKA protocols (when $m=50$)

Ratios of AKA protocols				Different values of “ n ”			
				300	500	800	1000
$PP_{(F-Auth)} / SE - AKA$	$PP_{(C-re-Auth)} / SE - AKA$	$PP_{(E-re-Auth)} / SE - AKA$	$PP_{(I-re-Auth)} / SE - AKA$	0.542-0.382 0.044-0.404	0.514-0.379 0.028-0.421	0.497-0.377 0.017-0.431	0.491-0.376 0.014-0.434
$PP_{(F-Auth)} / EG - AKA$	$PP_{(C-re-Auth)} / EG - AKA$	$PP_{(E-re-Auth)} / EG - AKA$	$PP_{(I-re-Auth)} / EG - AKA$	0.521-0.367 0.043-0.389	0.500-0.368 0.027-0.409	0.486-0.369 0.017-0.421	0.482-0.369 0.014-0.426
$PP_{(F-Auth)} / NOVEL - AKA$	$PP_{(C-re-Auth)} / NOVEL - AKA$	$PP_{(E-re-Auth)} / NOVEL - AKA$	$PP_{(I-re-Auth)} / NOVEL - AKA$	0.805-0.567 0.066-0.601	0.792-0.584 0.043-0.649	0.784-0.595 0.028-0.680	0.781-0.598 0.023-0.691
$PP_{(F-Auth)} / GROUP - AKA$	$PP_{(C-re-Auth)} / GROUP - AKA$	$PP_{(E-re-Auth)} / GROUP - AKA$	$PP_{(I-re-Auth)} / GROUP - AKA$	0.414-0.292 0.034-0.309	0.404-0.297 0.022-0.330	0.397-0.301 0.014-0.344	0.394-0.302 0.011-0.349
$PP_{(F-Auth)} / GBAAM - AKA$	$PP_{(C-re-Auth)} / GBAAM - AKA$	$PP_{(E-re-Auth)} / GBAAM - AKA$	$PP_{(I-re-Auth)} / GBAAM - AKA$	0.279-0.197 0.023-0.208	0.263-0.194 0.014-0.216	0.254-0.192 0.009-0.220	0.244-0.187 0.007-0.216
$PP_{(F-Auth)} / PRIVACY - AKA$	$PP_{(C-re-Auth)} / PRIVACY - AKA$	$PP_{(E-re-Auth)} / PRIVACY - AKA$	$PP_{(I-re-Auth)} / PRIVACY - AKA$	0.410-0.289 0.034-0.306	0.397-0.292 0.021-0.325	0.388-0.294 0.014-0.336	0.385-0.295 0.011-0.340
$PP_{(F-Auth)} / GLARM - AKA$	$PP_{(C-re-Auth)} / GLARM - AKA$	$PP_{(E-re-Auth)} / GLARM - AKA$	$PP_{(I-re-Auth)} / GLARM - AKA$	0.447-0.315 0.037-0.333	0.432-0.318 0.023-0.354	0.423-0.321 0.015-0.367	0.420-0.321 0.012-0.371
$PP_{(F-Auth)} / GR - AKA$	$PP_{(C-re-Auth)} / GR - AKA$	$PP_{(E-re-Auth)} / GR - AKA$	$PP_{(I-re-Auth)} / GR - AKA$	0.913-0.643 0.073-0.681	0.906-0.667 0.049-0.742	0.901-0.683 0.032-0.781	0.899-0.689 0.026-0.795
$PP_{(F-Auth)} / GBS - AKA$	$PP_{(C-re-Auth)} / GBS - AKA$	$PP_{(E-re-Auth)} / GBS - AKA$	$PP_{(I-re-Auth)} / GBS - AKA$	0.490-0.345 0.040-0.365	0.462-0.340 0.025-0.378	0.445-0.338 0.016-0.386	0.440-0.337 0.012-0.389
$PP_{(F-Auth)} / SEGB - AKA$	$PP_{(C-re-Auth)} / SEGB - AKA$	$PP_{(E-re-Auth)} / SEGB - AKA$	$PP_{(I-re-Auth)} / SEGB - AKA$	0.536-0.378 0.044-0.400	0.523-0.385 0.028-0.428	0.515-0.390 0.018-0.446	0.512-0.392 0.015-0.452

As can be seen, $PP_{(F-Auth)}$, $PP_{(C-re-Auth)}$, $PP_{(E-re-Auth)}$, and $PP_{(I-re-Auth)}$ are proposed protocol in the first authentication, complete re-authentication, external re-authentication, and internal re-authentication respectively. In the table, there are four columns to specify different values of “ n ” when $n = \{300, 500, 800, \text{ and } 1000\}$. In each column, we have 4 calculated numbers which are $\frac{PP_{(F-Auth)}}{AKA-Protocol}$, $\frac{PP_{(C-re-Auth)}}{AKA-Protocol}$, $\frac{PP_{(E-re-Auth)}}{AKA-Protocol}$ and $\frac{PP_{(I-re-Auth)}}{AKA-Protocol}$ where $AKA - Protocol$ is one of the other methods. The evaluation results show that the *TBC* ratios are decreasing as number of IoTDS (i.e., “ n ”) increasing. The results prove that the achieved average improvements of the $PP_{(F-Auth)}$, $PP_{(C-re-Auth)}$, $PP_{(E-re-Auth)}$, and $PP_{(I-re-Auth)}$ over the SE-AKA protocol are 48%, 62%, 97%, and 57% respectively. In Table 40, the average improvements of the REPS-AKA5 methods over all mentioned protocols are illustrated.

Table 40: The average improvements (TBC) of the REPS-AKA5

<i>AKA protocols compare to:</i>	<i>PP (F-Auth)</i>	<i>PP (C-re-Auth)</i>	<i>PP (E-re-Auth)</i>	<i>PP (I-re-Auth)</i>
SE-AKA	48%	62%	97%	57%
EG-AKA	50%	63%	97%	58%
NOVEL-AKA	20%	41%	96%	34%
GROUP-AKA	59%	70%	91%	66%
GBAAM-AKA	74%	80%	98%	78%
PRIVACY-AKA	60%	70%	93%	67%
GLARM-AKA	56%	68%	97%	64%
GR-AKA	9%	32%	95%	25%
GBS-AKA	54%	66%	97%	62%
SEGB-AKA	47%	61%	97%	56%

Another comparison of the group-based AKA protocols is shown in Figure 86 ((a), (b), (c), and (d)). The evaluation is based on different value of “ m ” when $m = \{30, 50, 80, 100\}$ and different number of IoT (i.e., “ n ” values). The first authentication, complete re-authentication, external re-authentication, and internal re-authentication phases are specified by REPS-AKA5 (F), REPS-AKA5 (C), REPS-AKA5 (E), and REPS-AKA5 (I) respectively. The results shows that the superiority of our protocol becomes more clear when the number of IoT group (i.e., “ m ” values) increases.

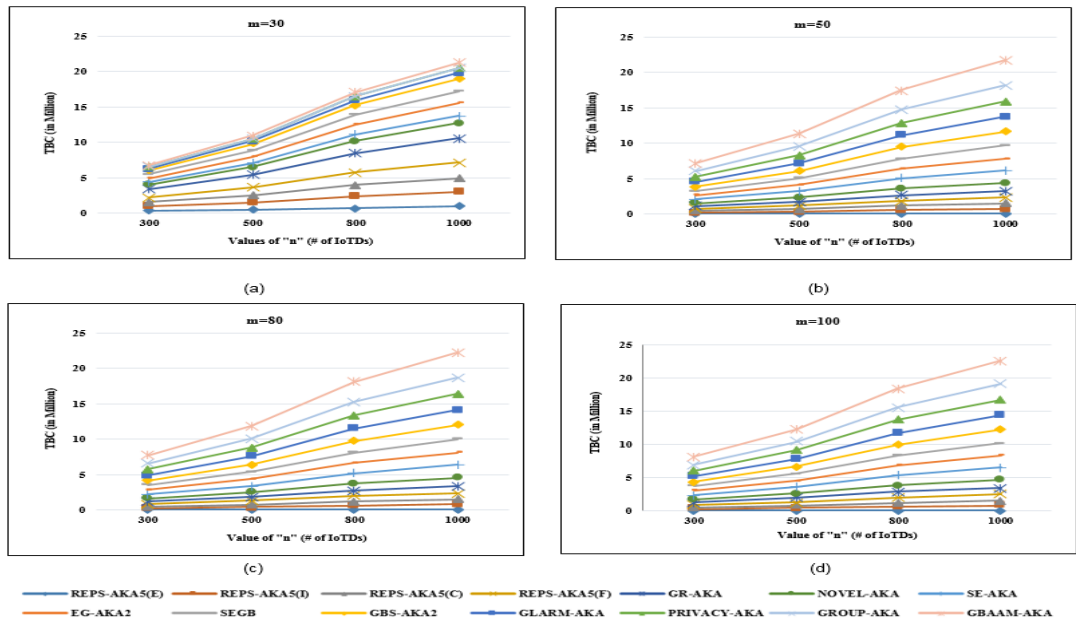


Figure 86: The evaluation results of TBC for group-based AKA protocols

7.4.2 Bandwidth consumption (core traffic)

We evaluate bandwidth consumption between MME and HSS nodes using a basic mobility model. In this model, the cost of traffic stream is calculated according to Mean Density (MD) of users in each cell, the speed (V) of the users, and length of the registration area (L). The assumption values for this model are presented in Table 41. The UE's can move in all directions $[0-2\pi]$. Hence, the Rate of authentication request (R) and Total Rate (TR) are considered by (7.2) and (7.3) respectively as [12, 24, 41]:

$$R = (MD \times V \times L) / \pi \quad (7.2), \quad TR = R \times Ar \quad (7.3)$$

Table 41: Assumption values (traffic flow)

Parameters of mobility model	Assumption values
Ar	100
L	25.3 km
MD	500 / km ²
V	4.8 km/h

Where “ Ar ” is number of registration areas. Bandwidth consumption (BW_{mh}) of core network is calculated according to the number of bits transmitted between MME and HSS. The BW_{mh} can be expressed as [12]: $BW_{mh} = \sum_{k=1}^{k=i} (MSG)_k \times TR \quad (7.4)$

Where “ i ” is the number of transmitted messages during the AKA procedure (between MME and HSS nodes), and $(MSG)_k$ is the length of k^{th} message (in bits). Thus, we evaluate the BW_{mh} of the REPS-AKA5 protocols and compare it with the standard EPS-AKA protocol and some other AKA methods as follows [12, 24, 41]:

The BW_{mh} of EPS-AKA: In the EPS-AKA protocol, two authentication messages are used between the MME and the HSS. The MME in the serving network transmits a request to the HSS via $MSG1$. In response, the HSS in home network forwards “ a ”

number of AVs to the MME via *MSG2* message. Therefore, the number of transmitted bits can be presented as: $\sum_{k=1}^{k=2}(MSG)_k = MSG_1 + MSG_2 = 176 + 544a$. We use equation (7.4) with the parameters in Table 41. We assume that the number of AVs is 10 (i.e., $a = 10$) and there are 500 IoTs formed as 30 IoT groups (i.e., $n=500$, $m=30$). Thus, the bandwidth consumption of core network (i.e., BW_{mh}) can be found as 3.01 Mbits/s.

The BW_{mh} of the Modified LTE-AKA (MLTE-AKA): In the MLTE-AKA method, messages *MSG1* and *MSG2* are forwarded between involved nodes. Therefore, the number of transmitted bits in the EPC part of the LTE network for MLTE-AKA can be calculated as: $\sum_{k=1}^{k=2}(MSG)_k = MSG_1 + MSG_2 = 176 + 800a$. Thus, using equation (7.4) with the assumption values (Table 41), and if the number of AVs is 10 (i.e., $a = 10$), the BW_{mh} for the MLTE-AKA can be found as 4.39 Mbits/s (when $n=500$, $m=30$).

The BW_{mh} of Enhanced-AKA (E-AKA): In the E-AKA protocol, messages *MSG1* and *MSG2* are shared in the core network. The MME requests security parameters from the HSS by sending *MSG1*. And the HSS forwards AVs to the MME via *MSG2*. The number of transmitted bits in the core network during this AKA procedure can be found as: $\sum_{k=1}^{k=2}(MSG)_k = MSG_1 + MSG_2 = 512 + 384 = 896$. Hence, by using (7.4) and Table 41 (when $n=500$, $m=30$), the BW_{mh} of the E-AKA will be 0.48 Mbits/s.

The BW_{mh} of Efficient AKA-EPS (EAKA-EPS): In this method, messages shared in the core network (i.e., between HSS and MME) for AKA procedure are *MSG1* and *MSG2*. Thus, the number of transmitted bits between MME and HSS can be presented as: $\sum_{k=1}^{k=2}(MSG)_k = MSG_1 + MSG_2 = 352 + 384 = 736$. Based on (7.4) and Table 41, if $n=500$ and $m=30$, the BW_{mh} of the EAKA-EPS will be 0.39 Mbits/s.

The BW_{mh} of REPS-AKA5: In REPS-AKA5, we require two messages (i.e., MSG_1 and MSG_2) shared between MME and HSS in EPC part of LTE network. The total number of transmitted bits between HSS and MME can be found as: $\sum_{k=1}^{k=2}(MSG)_k = MSG_1 + MSG_2 = 256 + 544 = 800$. Thus, using (7.4) with the parameters in Table 41 (when $n=500$, $m=30$), the BW_{mh} of the REPS-AKA5 can be found 0.02 Mbits/s.

Accordingly, we compared the BW_{mh} of the AKA protocols as shown in Figure 87 (where $a=10$, $n=300$, and $m=100$).

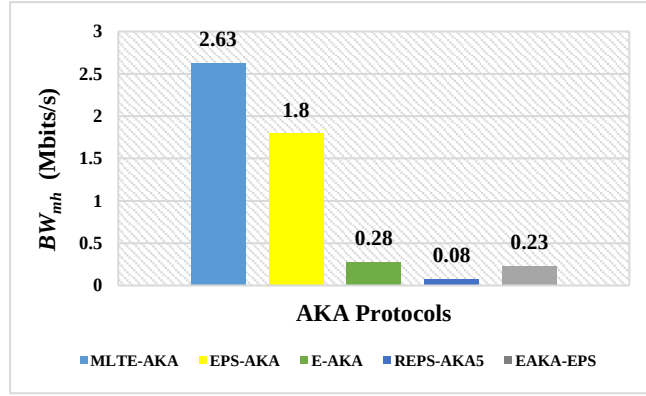


Figure 87: The evaluation results of BW_{mh} of AKA protocols

The evaluation results show that the achieved improvements of the REPS-AKA5 protocol over the EPS-AKA, E-AKA, MLTE-AKA, and EAKA-EPS are 95%, 71%, 96%, and 65% respectively. In Figure 88 ((a), (b), (c), and (d)), a comparison between these AKA methods is shown based on different AV sizes (i.e., $a = \{5, 10, 15, 20\}$), and the MD of IoTDS. The MD is a combination of value “ n ” and “ m ” which are total number of IoTDS and number of IoT groups respectively. As expected, the BW_{mh} of all the AKA protocols increases as MD increases. However, the REPS-AKA5 protocol outperforms the other methods as it has the lowest BW_{mh} and the smallest slope. In

addition, the superiority of REPS-AKA5 protocol becomes clearer when the number of AVs (i.e., “ a ” values) increases.

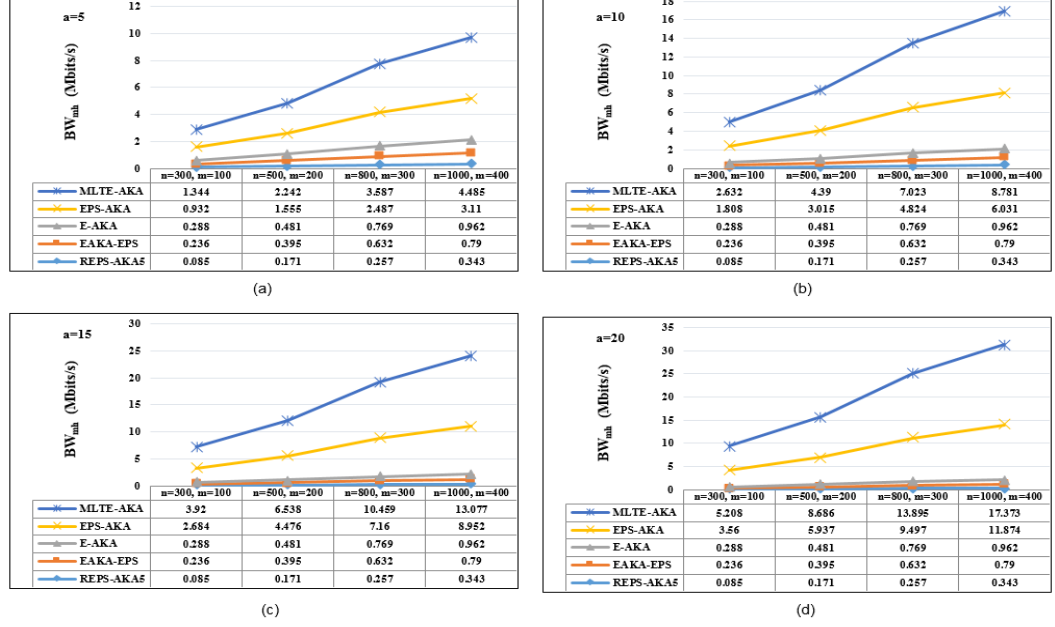


Figure 88: The BW_{mh} of AKA protocols for different AVs

7.4.3 Computational cost

To compute the Computational Cost (CC), we evaluate security functions that are used in the AKA methods. Accordingly, the CC of AKA protocol (CC_{AKA}) is presented for the UE (IoT), MME, and the HSS based on the performed security functions by using equation (7.5) below [12, 24, 41]: $CC_{AKA} = \sum_{k=1}^{i=3} Cost(node)_k \times TR$ (7.5)

Where “ i ” is the number of nodes (i.e., $i = 3$: IoT, MME, and HSS) in the system. In this section, we compare the CC_{AKA} of the REPS-AKA5 method with some other AKA methods as follows [12]:

The CC of the EPS-AKA: Computational cost of security functions in the EPS-AKA scheme are as follows: $UE(IoTD)_{cost} = (f1 + f2 + f3 + f4 + f5 + 2KDF) = 7$, $MME_{cost} =$

(No function) = 0, $HSS_{cost} = (f1 + f2 + f3 + f4 + f5 + 2KDF)a = 7a$, Where “a” is the number of AVs. Hence, by using equation (7.5), the CC of the EPS-AKA ($CC_{EPS-AKA}$) can be found as: $CC_{EPS-AKA} = TR \times (7 + 7a)$

The CC of the MLTE-AKA: In the MLTE-AKA protocol, the CC of the used functions in the involved entities (i.e., IoT, MME, and HSS) are as follows:

$UE(IoTD)_{cost} = (Enc + 2fk + MAC + f1 + f2 + f3 + f4 + f5 + 3KDF) = 12$, $MME_{cost} = (No\ function) = 0$, $HSS_{cost} = Dec + (3KDF + f1 + f2 + f3 + f4 + f5)a = 1 + 8n$, Where “a” is the number of AVs. Accordingly, using equation (7.5), the CC of the MLTE-AKA ($CC_{MLTE-AKA}$) is: $CC_{MLTE-AKA} = TR \times (13 + 8a)$

The CC of the EAKA-EPS: Computational cost related to security functions in the EAKA-EPS method are as follows: $UE(IoTD)_{cost} = (2Enc + 2MAC + f1 + f2 + f3 + f4 + f5 + 2KDF + Dec) = 12$, $MME_{cost} = (Enc + MAC + f1 + f2 + f3 + f5 + KDF + 2Dec) = 9$, $HSS_{cost} = (f3 + f4 + KDF) = 3$. By using equation (7.5), the computation cost of the EAKA-EPS ($CC_{EAKA-EPS}$) can be found as: $CC_{EAKA-EPS} = TR \times (24)$

The CC of the E-AKA protocol: Total number of security functions used in the REPS-AKA5 protocol are as follows: $UE(IoTD)_{cost} = (f' + f1 + f2 + f3 + f4 + 2f5 + KDF + Dec + Enc) = 10$, $MME_{cost} = (f1 + f2 + f3 + f4 + f5 + KDF + Dec + Enc) = 8$, $HSS_{cost} = (f' + KDF) = 2$. For each AKA request, the number of security functions performed in E-AKA method is 20. Hence, the computation cost of the E-AKA (CC_{E-AKA}) is: $CC_{E-AKA} = TR \times (20)$

The CC of the REPS-AKA5: Computational cost of security functions in the REPS-AKA5 protocol are as follows: $UE(IoTD)_{cost} = ((f1 + f6 + f9 + KDF)n + (f1 + f2 +$

$f3 + f4 + f5 + f8 + f9 + 4KDF + AuthFunc)m) = 4n + 13m, \mathbf{MME}_{cost} = (NIFunc + AuthFunc) = 2, \mathbf{HSS}_{cost} = (f1 + f2 + f3 + f4 + f5 + f6 + f7 + f8 + 3KDF + NIFunc) = 12,$
 Where “ n ” and “ m ” are number of IoT devices and the number of IoT groups respectively.
 Hence, using (7.5) the total computation cost of the REPS-AKA5 protocol ($CC_{REPS-AKA5}$) can be calculated as: $CC_{REPS-AKA5} = TR \times (4n + (13m + 14))$

We calculate CC_{AKA} of the mentioned protocols when number of AV is 10, $n=300$, and $m=100$. In addition, we assumed other parameters based on Table 41. Accordingly, the computation costs of AKA protocols are presented in Figure 89.

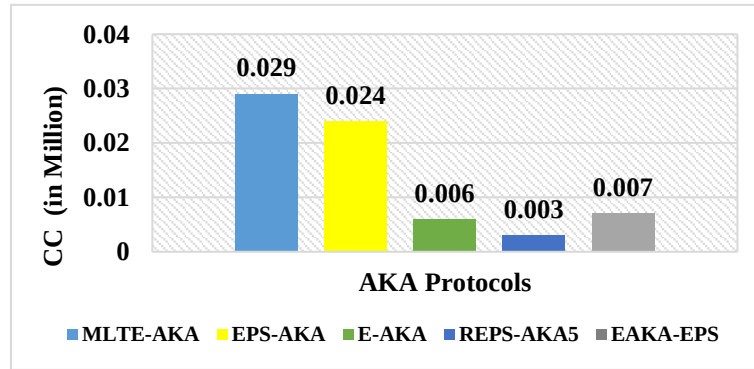


Figure 89: Computational cost in AKA protocols

As it can be seen from Figure 89, the CC improvements of REPS-AKA5 over the standard AKA protocol (EPS-AKA), E-AKA, MLTE-AKA, and EAKA-EPS are 87%, 50%, 89%, and 57% respectively. Figure 90 ((a), (b), (c), and (d)), shows the CC of various protocols based on different numbers of IoT device (i.e., “ n ” and “ m ” values) and AVs (i.e., “ a ” value). The results show that the proposed AKA method is much better than other security protocols, especially when the number of IoT devices increases. Moreover, the superiority of the REPS-AKA5 method becomes more clear when the number of AVs increases.

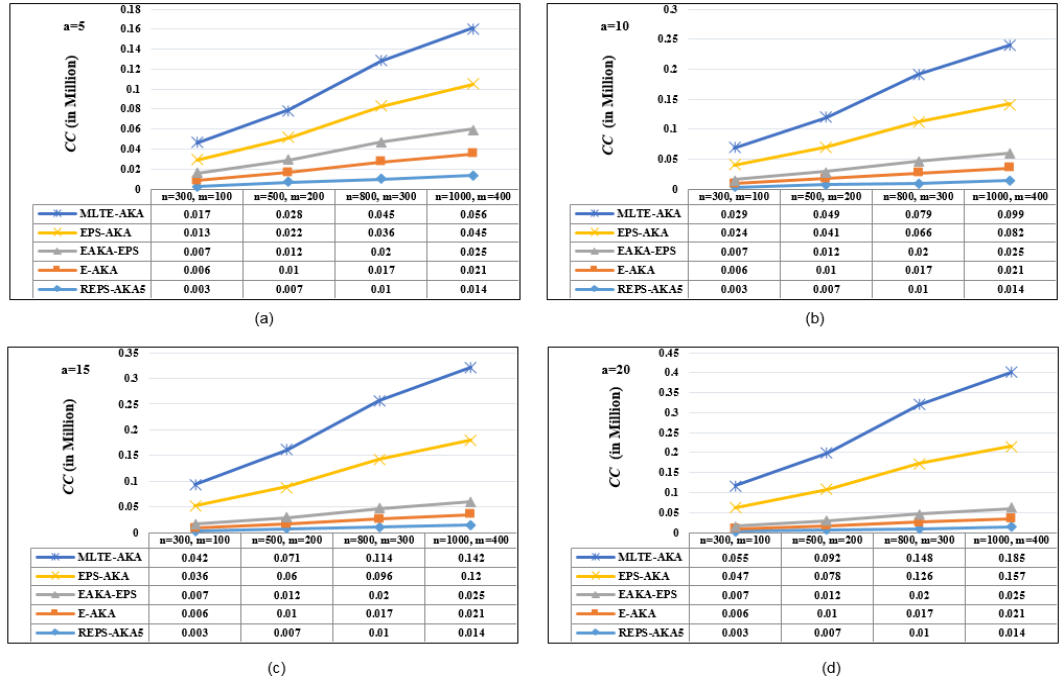


Figure 90: The CC of AKA protocols for different AVs

In addition, we evaluate the computation cost of the AKA protocols based on involved operation. The delay time of involved operations are considered including the hash function (DT_{hash}), multiplication over the elliptic curve (DT_{mul}), XOR operation (DT_{XOR}), symmetric AES encryption (DT_{AES}), asymmetric DSA encryption (DT_{DSA}), modulus (DT_{mod}), random number generation (DT_{Rand}), and Lagrange component time at IoTD (DT_L). Moreover, other operations are ignored, as they are low cost. We assume that the delay time of mentioned operation are based on Table 42 (using crypto++ library) [9].

Table 42: Cost (Time delay) of the different operations

Operation	Symbol	Delay Time (μs)
hash function (SHA-256)	DT_{hash}	67
AES encryption	DT_{AES}	161
random number generation (128)	DT_{Rand}	45
XOR operation	DT_{XOR}	2
Lagrange component time	DT_L	57
DSA encryption	DT_{DSA}	280
multiplication over the elliptic curve	DT_{mul}	612
Modulus	DT_{mod}	124

We calculate the cost of the operations for each AKA protocol separately. For the evaluation of delay time (i.e., Computational cost), we suppose that there are “ n ” number of IoT devices, which form “ m ” IoT groups. In Table 43, the used equations to compute total delay time are presented (where a =number of authentication vectors) [9] [52] [53] [56] [60]. We named first authentication, complete re-authentication, external re-authentication, and internal re-authentication phases by REPS-AKA5 (F-Auth.), REPS-AKA5 (C-r-Auth.), REPS-AKA5 (E-r-Auth.), and REPS-AKA5 (I-r-Auth.) respectively.

Table 43: Total delay time of the different AKA protocols

Protocol name	Total delay time
EPS-AKA	$n(5DT_{hash} + DT_{XOR} + DT_{Rand} + a(6 DT_{hash} + DT_{XOR}))$
EG-AKA	$n(10DT_{hash} + 2DT_{Rand} + 3DT_{mul} + 2 DT_{AES}) + m(8 DT_{hash} + 2 DT_{Rand} + 3 DT_{mul} + 2 DT_{AES})$
GBAAM	$n(4DT_{hash} + DT_{XOR} + DT_{Rand} + 2 DT_{AES} + a(6 DT_{hash} + DT_{XOR}))$
E²SEC	$n(4DT_{hash} + DT_{XOR} + DT_{Rand} + a(6 DT_{hash} + DT_{XOR})) + 2 DT_{mod} + DT_{hash} + DT_{AES}$
Fu's scheme	$n(11DT_{hash} + 3DT_{Rand} + 4DT_{mul} + 6 DT_{XOR}) + m(4DT_{hash} + 2 DT_{Rand} + 2DT_{mul} + DT_{XOR})$
GR-AKA	$n(DT_L + 2 DT_{mul} + DT_{hash}) + m(5DT_{hash} + DT_L + 2DT_{mul})$
PRIVACY-AKA	$n(8DT_{hash} + 3DT_{mul}) + m(6DT_{hash} + DT_{mul})$
REPS-AKA5 (F-Auth.)	$n(2DT_{XOR} + 5DT_{hash} + DT_{Rand} + 3 DT_{DSA}) + m(8DT_{XOR} + 19DT_{hash} + 2 DT_{AES} + 2DT_{Rand} + 2 DT_{DSA})$
REPS-AKA5 (C-r-Auth.)	$n(2DT_{XOR} + 4DT_{hash} + 2 DT_{DSA} + DT_{Rand}) + m(9DT_{XOR} + 10DT_{hash} + 4 DT_{DSA} + 2DT_{Rand})$
REPS-AKA5 (E-r-Auth.)	$m(10DT_{XOR} + 10DT_{hash} + 3 DT_{DSA} + 2DT_{Rand})$
REPS-AKA5 (I-r-Auth.)	$n(5DT_{XOR} + 8DT_{hash} + 3 DT_{DSA} + 2DT_{Rand})$

We analyzed the delay time for some other AKA protocols corresponding to the different values of “ n ” and “ m ” as shown in Figure 91 (where $a=7$). As can be seen, the total delay time of proposed REPS-AKA5 protocol is less than other AKA methods including EPS-AKA which is standard AKA method in LTE system. The evaluation results show that the superiority of proposed REPS-AKA5 protocol becomes clearer when the number of IoT groups and IoT devices (i.e., “ m ” and “ n ” values) increase. In addition, the presented method in re-Authentication phases (i.e., REPS-AKA5 (C-r-

Auth.), REPS-AKA5 (E-r-Auth.), and REPS-AKA5 (I-r-Auth.)) has less computational cost (i.e., Total delay time) because system can employ saved security keys/parameters.

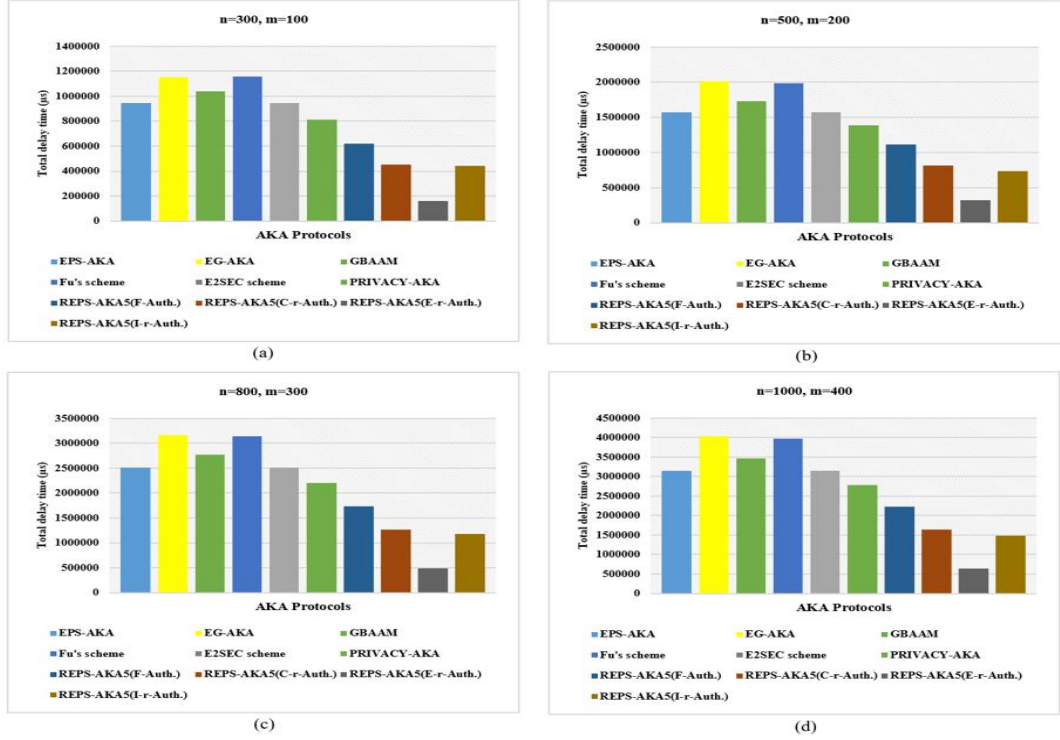


Figure 91: The CC of Group-based AKA protocols for different AVs

Moreover, there are some methods which have better CC (e.g., NOVEL-AKA[59]= $n(6DT_{hash})+m(2DT_{hash})+DT_{hash}$, GLARM-AKA[71]= $n(7DT_{hash})+m(4DT_{hash})$). However, according to other performance metrics and security evaluations, the REPS-AKA5 is much better than them.

7.4.4 Signaling overhead

It is valuable to present a comparative analysis of the Signaling Overhead (SO). We compare the SO of the presented REPS-AKA5 with other AKA methods. For the analysis of the SO, we consider that there are “n” IoTs and “m” IoT groups (where $n>m$). The evaluation results of the SO of AKA protocols are demonstrated in Table

44. In addition, we analyze the *SO* of AKA protocols based on different number of IoT_D (i.e., “*n*”) and IoT groups (i.e., “*m*”) as shown in Figure 92 [9] [12].

Table 44: Signaling overhead of the AKA protocols

AKA protocols	SO	AKA protocols	SO
EPS-AKA	$6n$	PRIVACY-AKA	$6n+2m$
SE-AKA	$6n+2m$	GLARM-AKA	$6n+2m$
EG-AKA	$10n+2m$	GR-AKA	$6n+2m$
NOVEL-AKA	$6n+2m$	GBS-AKA	$6n+2m$
GROUP-AKA	$6n+3m$	SEGB-AKA	$6n+2m$
GBAAM-AKA	$6n+4m$	REPS-AKA5	$3n+2m$

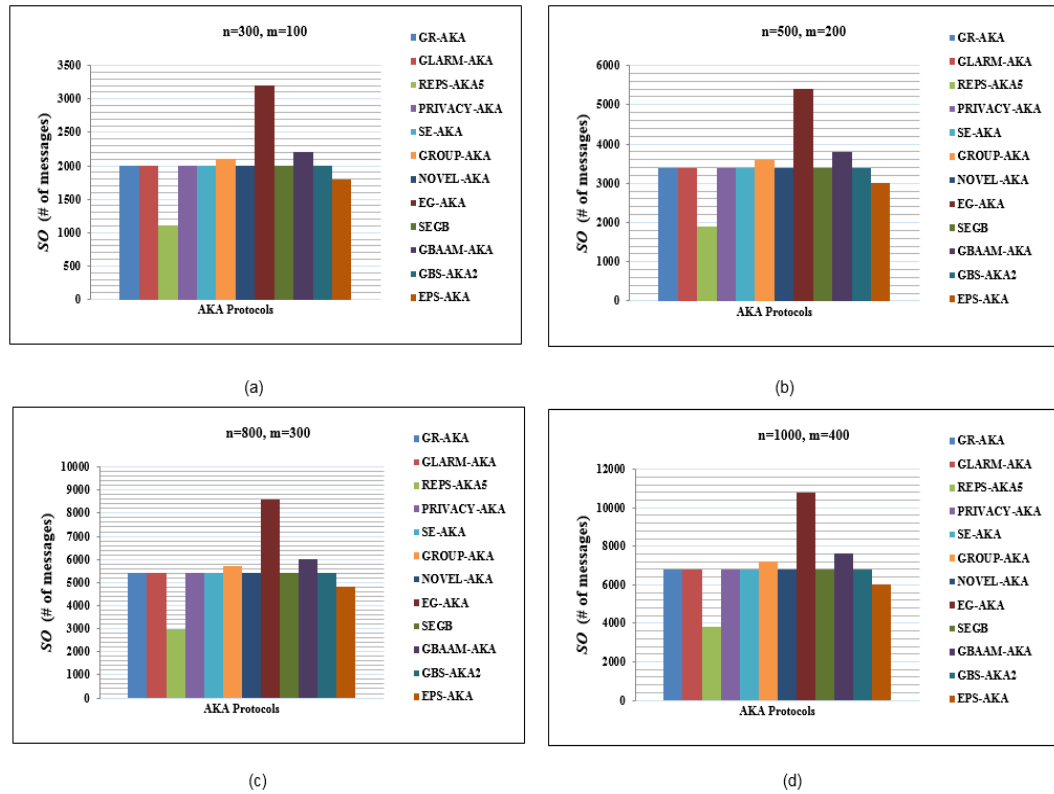


Figure 92: The signaling overhead of AKA protocols for different “*n*” and “*m*”

According to Figure 92, the REPS-AKA5 outperforms the others as it has the lowest *SO*. Moreover, the superiority of REPS-AKA5 becomes clearer when we increase the number of IoT_Ds and IoT groups. The average *SO* improvements of REPS-AKA5 over EPS-AKA, SE-AKA, EG-AKA, NOVEL-AKA, GROUP-AKA, GBAAM-AKA,

PRIVACY-AKA, GLARM-AKA, GR-AKA, GBS-AKA, and SEGB-AKA are 37%, 44%, 65%, 44%, 47%, 50%, 44%, 44%, 44%, 44%, and 44% respectively.

7.4.5 Storage overhead in the serving network

The Storage Overhead (StO) of the AKA protocols are analyzed based on the number of stored bits in the MME. In the REPS-AKA5, the MME uses its generated UPK to calculate fresh values including $NKASME_m$, $NAUTH_m$, and $NXRESL_m$. Thus, the MME needs only one AV for AKA procedure. The comparison of the StO is shown in Table 45, where “ a ” is the number of the AVs [10, 12].

Table 45: Storage overhead of AKA methods

protocols	StO (bits)	protocols	StO (bits)	protocols	StO (bits)
Ref. [29]	$432a$	Ref. [62]	$608a$	Ref. [25]	$800a$
Ref. [24]	$640a$	Ref. [63]	$704a$	REPS-AKA5	512
Ref. [10]	$392a$				

Comparative results present that REPS-AKA5 reduces the StO considerably. An evaluation of StO of different AKA methods is demonstrated in Figure 93 (when $a=5$). The StO improvements of REPS-AKA5 over Ref [24], Ref [29], Ref [10], Ref [62], Ref [63], and Ref [25] are 84%, 77%, 73%, 83%, 85%, and 87% respectively.

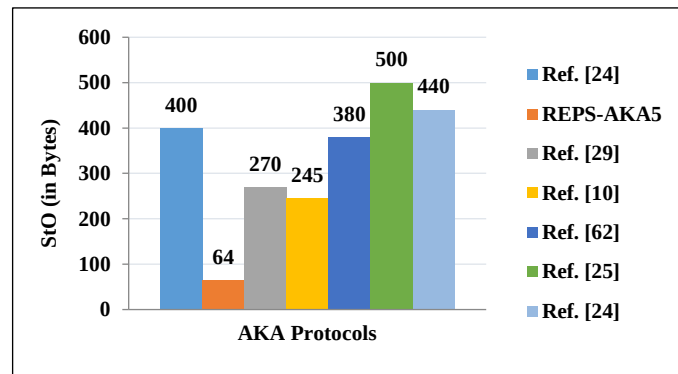


Figure 93: Storage overhead in AKA protocols (AV=5)

In addition, the storage overheads of group-based AKA protocols are compared with our proposed REPS-AKA5 protocol as shown in Table 46 [42] [9], where “ n ” and “ m ” are total number of IoTDs and number of IoT groups respectively.

Table 46: Storage overhead of group-based protocols

AKA protocols	StO (bits)	AKA protocols	StO (bits)
<i>NOVEL-AKA</i>	816m	<i>GLARM-AKA</i>	1072m
<i>GROUP-AKA</i>	1200m	<i>SEGB-AKA</i>	688m
<i>GBAAM-AKA</i>	128n+736m	<i>REPS-AKA5</i>	512m
<i>PRIVACY-AKA</i>	992m		

We compare storage overheads of these AKA protocols (group-based method) for different values of “ n ” and “ m ” as shown in Figure 94. The average “storage overhead” improvements of REPS-AKA5 protocol over NOVEL-AKA, GROUP-AKA, GBAAM-AKA, PRIVACY-AKA, GLARM-AKA, and SE-AKA are 37%, 57%, 52%, 48%, 52%, and 25% respectively.

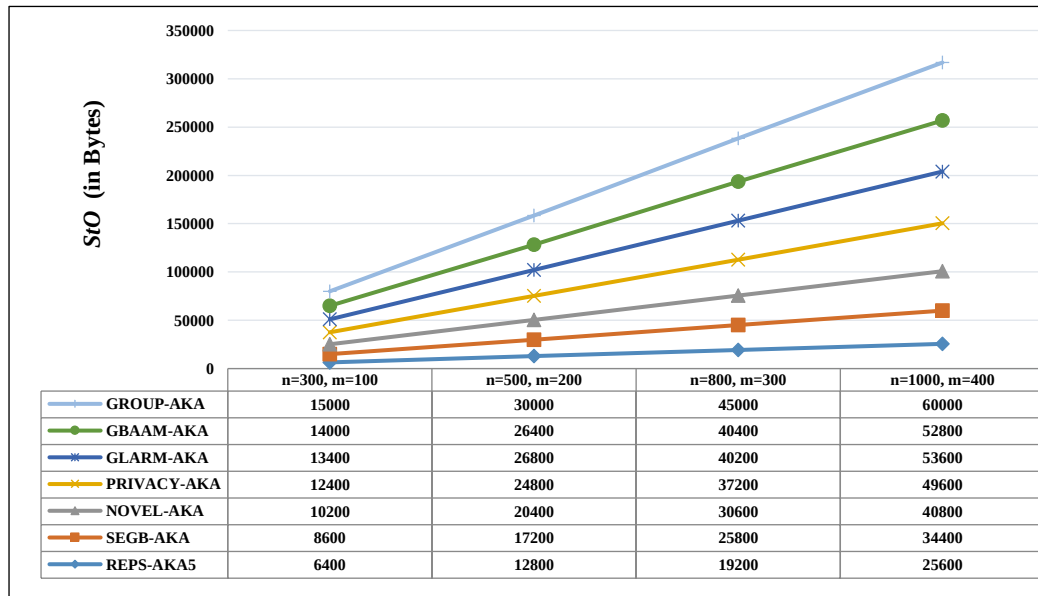


Figure 94: Storage overhead in AKA protocols (group-based)

Chapter 8

CONCLUSION

In the LTE networks, many security vulnerabilities may threaten the users. The DoS and MitM attacks are two examples to show the need for strong and trustable AKA protocol. Consequently, authentication is the most significant step in securing LTE systems and must be established properly. We presented a Robust AKA protocol (i.e., REPS-AKA1) to cover user confidentiality and mutual authentication in chapter 3. The REPS-AKA1 can protect the LTE system against the security vulnerabilities such as IMSI leakage, issues of user tracking, DoS attacks, and MitM attacks. Compared with the EPS-AKA protocol and some suggested AKA methods, the proposed protocol improves the computational cost, bandwidth consumption, and signaling overhead significantly. According to the security evaluation in AVISPA tool, the REPS-AKA1 protocol is safe in both OFMC, and CL-AtSe modes. Accordingly, we can say that the REPS-AKA1 method accomplishes the secrecy and authentication goals in the LTE networks. In addition, the REPS-AKA1 is verified using Scyther tools and BAN Logic.

Then, we proposed a robust scheme (REPS-AKA2) for X2-handover by protecting key parameters in chapter 4. The REPS-AKA2 scheme can provide the authentication features without involving the home network (i.e. HSS), which is a key parameter to improve efficiency in AKA methods. Moreover, impersonating involved nodes in the REPS-AKA2 protocol is infeasible because all transmitted messages are signed by strong security keys. We need the security parameter of all involved nodes (UE, source

eNodeB, and target eNodeB) to generate NKG and UK. This feature makes more robust keys than other methods that compute the main keys by only one involved node, since the responsible node may select weak parameters to generate main keys. Besides, the proposed scheme can provide message authentication in the S1 interface for MME using the MVP parameter. The security analysis is checked by using BAN Logic, AVISPA, and Scyther tools to demonstrate that the REPS-AKA2 protocol thwarts well-known attacks. Moreover, the suggested handover approach needs fewer number of transmitted packets and communication overloads. Total communication overload is 6 and 3 in the first (Initial Authentication) and second (re-Authentication) phases respectively. Thus, the communication overload is much less than the original X2 handover and other schemes. Moreover, the nodes (i.e. UEs, BSs and MMEs) have to send much less packets in Initial Authentication (476 bytes) and re-Authentication (196 bytes) phases. Consequently, the proposed protocol has higher security level and acceptable performance (QoS level) when compared to other AKA schemes.

In chapter 5, we proposed a performance-enhanced AKA method for LTE networks. The proposed protocol can solve most of the security issues that still occur in the EPS-AKA protocol. In this REPS-AKA3, involved entities (i.e., UE, eNodeB, MME, and HSS) strongly authenticate each other. Security services included in the proposed REPS-AKA3 are mutual authentication, improved NAS key management, and enhanced AS key agreement. The AS and NAS keys, including KRRCCint, KRRCCenc, KRRCCUPenc, KNASenc, and KNASint, are based on calculated NKG and UK in first authentication and re-authentication respectively. The REPS-AKA3 protocol provides perfect privacy for the LTE entities appropriately. The proposed scheme completely follows the architecture of the LTE systems and avoids extra signaling overheads.

Thus, it is adaptable to the current LTE systems. The security analysis of the proposed REPS-AKA3 protocol is performed using BAN Logic, Scyther, and AVISPA tools. The evaluation results prove the security capabilities of the REPS-AKA3 method against various malicious attacks such as replay attacks, redirection attacks, and MitM attacks. Moreover, we compare the REPS-AKA3 protocol with other existing AKA approaches in terms of bandwidth consumption, computational cost, signaling overhead, and storage overhead. These parameters in all protocols increase as MD increases, however, the proposed REPS-AKA3 outperforms the others as it has the lowest bandwidth consumption, computational cost, and storage overhead compared with other protocols. The superiority of our protocol becomes outstanding when the number of AV increases since the REPS-AKA3 performance is independent of number of AVs. Thus, achieved results indicate that the REPS-AKA3 protocol can guarantee an acceptable performance while satisfying more security goals (e.g. strong mutual authentication, and appropriate secrecy) in LTE networks.

Authentication and key agreement are two very important steps in securing LWI systems. Thus, these steps must be implemented correctly to prevent malicious attacks. The importance of AKA phase rises when UE moves between wireless networks (e.g. LTE and WLAN) since they are implemented based on various security layers. The standard protocol in LTE (i.e., EPS-AKA) and in LWI (i.e., Full and Fast EAP-AKA') do not provide acceptable protection for LWI systems and these networks still suffer from some security threats such as DoS/DDoS attacks, and MitM attacks. These attacks occur because the identity of user (i.e., IMSI value) is transmitted in clear text during standard AKA procedure. In chapter 6, we proposed a set of secure and optimized AKA protocols for first attachment, and n^{th} attachment during handover

between the LTE and the WLAN systems. The proposed REPS-AKA4 protocols improve security level of the LWI systems without compromising the level of QoS. We provide a combination of public and secret key infrastructure to satisfy both security and QoS requirements appropriately. In the REPS-AKA4 protocols, different re-authentication methods for user attachment, inter-handover and intra-handover in LWI system were adopted. The proposed REPS-AKA4 protocols are evaluated according to important metrics including computational cost, bandwidth consumption, authentication delay, and signaling cost. Compared with the Full and Fast EAP-AKA' protocols, the proposed protocols have better authentication delay, and signaling cost. In addition, the REPS-AKA4.ph1.1 and REPS-AKA4.ph1.2 outperform the other LTE-AKA protocols since they have the lowest computational cost and bandwidth consumption. The superiority of our protocols become clearer when the number of AV increases during AKA procedure. According to security evaluations, the proposed protocol satisfy the defined security goals properly. The results from AVISPA verification tools (in CL-AtSe and OFMC modes), Scyther tools, and BAN logic show that the suggested methods are resistant to active and passive attacks. Consequently, the suggested REPS-AKA4 protocols achieve robust security and better performance compared with the standard protocols including EPS-AKA, standard Full and Fast EAP-AKA' protocols.

In chapter 7, we focused on the security and performance issue of the IoTs in LTE-enabled IoT systems. We proposed a robust group-based AKA method named as REPS-AKA5 with an efficient update-policy. This protocol is able to provide a group authentication mechanism to verify IoT groups efficiently. The REPS-AKA5 approach suggested not only a mechanism for first authentication of the IoT groups, but also it

provided some efficient and practical techniques for the re-authentications phases. The re-authentications phase includes complete re-authentication, external re-authentication, and internal re-authentication. In these re-authentications phases, the previously generated parameters (i.e., $AUTH_m$, $XRESL_m$, and $KASME_m$) are used to produce new security parameters. The suggested method also uses fresh random values to satisfy freshness feature for calculated keys/parameters (i.e., $NKASME_m$, $NAUTH_m$, and $NXRESL_m$). The REPS-AKA5 protocol uses a combination of symmetric and asymmetric infrastructures to secure message transmission in the control plan. This hybrid structure not only makes our method safe, but also it optimizes the performance of the LTE-enabled IoT system properly. In the REPS-AKA5 method, the $IoTL_m$ performs a secure access policy for each IoT device by using GK_m , GK_m , and SK_{n-m} to provide Perfect Forward/Backward Secrecy. In group updating phase, when a new member joins the IoT group, $IoTL_m$ generates new/temporary security parameters (i.e., NDI_{n-m} and TDI_{n-m}) for the newly joined IoT device. Thus, the suggested method satisfies PBS/PFS in updating phase properly. Therefore, the REPS-AKA5 meets the security requirements including confidentiality in control plan, robust mutual authentication, strong message authentication, and PFS/PBS. The presented approach also resists several security threats such as replay attacks, MitM attack, DoS attack, redirection attack, and single-key dependency. The security of the presented approach is verified by AVISPA tool, Scyther tool, and BAN Logic. We also compared our method with other AKA methods based on some security properties such as privacy protection, resistance to MitM attack, and resistance to replay attack. The security evaluation results show that the REPS-AKA5 satisfies more security goals than other AKA methods. In addition, the presented approach outperforms other AKA protocols in terms of computational cost, storage overhead,

bandwidth consumption, and signaling overhead. Consequently, the REPS-AKA5 improves the security level of the IoT systems and simultaneously resolves the performance-based issues considerably.

As a summary, the REPS-AKA methods could address a set of the most significant security issues in LTE and LTE-enabled IoT systems. This group of AKA protocols is able to provide security requirements not only in LTE networks but also in LWI systems (i.e., REPS-AKA4). In addition, one type of these AKA methods (i.e., REPS-AKA2) is defined to handle X2-handover in LTE systems. Another one of the proposed AKA methods (REPS-AKA5) is designed to handle IoT systems. The REPS-AKA5 approach can be run by different IoT devices in a wide range of IoT applications, especially in BMS and Smart homes. So, the set of proposed AKA methods can satisfy security goals in LTE attachment, handover procedure, LWI systems, and LTE-enabled IoT applications. A summary of REPS-AKA protocols is shown in Figure 95.

Protocol Name	Performance analyze metrics	Security evaluation tools	For LTE network	For LTE- enabled IoT system (Mobile devices)	For LTE- enabled IoT network (all IoT devices)	For X2- handover in LTE systems	For LTE- WLAN systems	Pre- authentication procedure (or grouping policy)	Authentication	Key Agreement	Re- Authentication
REPS-AKA1	$BW_{nh}^{*}/CC/ SO/ StO$	AVISPA/Scyther/ BAN Logic	YES	YES					YES	YES (Master key)	
REPS-AKA2	$CO/TP/CC$	AVISPA/Scyther/ BAN Logic	YES	YES		YES			YES	YES (Master key, AS layer, NAS layer)	YES
REPS-AKA3	$BW_{nh}^{*}/TBC/CC/ SO/ StO$	AVISPA/Scyther/ BAN Logic	YES	YES				YES	YES	YES (Master key, AS layer, NAS layer)	YES
REPS-AKA4	$BW_{nh}^{*}/TBC/CC/AD/SC$	AVISPA/Scyther/ BAN Logic	YES	YES			YES		YES	YES (Master key)	YES
REPS-AKA5	$BW_{nh}^{*}/TBC/CC/ SO/ StO$	AVISPA/Scyther/ BAN Logic	YES	YES	YES			YES	YES	YES (Master key)	YES

Figure 95: Summary of the REPS-AKA protocols

In the future, Fifth-generation wireless (5G) networks will affect many network-based systems, especially IoT applications. Consequently, as a future work, we need to design an optimized AKA method that is well-matched with 5G networks as well.

Thus, designing an AKA protocol for IoT applications that is compatible with both LTE and 5G infrastructure is research worthy. Accordingly, it can be great if we improve REPS-AKA protocols to be adaptive to 5G networks. The AKA protocol is an essential security mechanism in both LTE and 5G networks to protect the privacy and integrity of user's (i.e., IoT) information. While the AKA protocol used in LTE and 5G networks has some similarities, there are also some differences in terms of the algorithm used, message formats, and procedures [107-110]. To adapt the AKA protocol used in LTE to the 5G network, the following steps can be taken:

1. We need to modify the REPS-AKA protocols to support the new security features of 5G systems, such as 5G-AKA, which is the new version of AKA used in 5G networks.
2. We need to update the message formats and procedures to support the new requirements and features of 5G networks. For example, in 5G networks, the security architecture is based on the Service-Based Architecture (SBA), which requires different security mechanisms than the LTE core network.
3. We should define an appropriate mapping of the new 5G-AKA security parameters to the REPS-AKA protocols to ensure backward compatibility with LTE networks.
4. We need to conduct thorough testing and verification of the adapted AKA protocol to ensure interoperability and security in 5G networks. We can prove its security by means of the AVISPA tool, Scyther tool, and BAN Logic.

Thus, adapting the REPS-AKA protocols used in LTE to 5G networks requires a comprehensive understanding of the differences between the two networks and the new security features and requirements of 5G networks.

REFERENCES

- [1] M. Adhikari, P. K. Panda, S. Chattopadhyay, and S. Majumdar, "A Novel Group-based Authentication and Key Agreement Protocol for IoT Enabled LTE/LTE–A Network," *International Conference on Wireless Communications Signal Processing and Networking*, pp. 168-172, 2020.
- [2] P. K. Panda, and S. Chattopadhyay, "An Enhanced Secure Authentication and Key Agreement Scheme for LTE Networks," *3rd International Conference on Contemporary Computing and Informatics*, pp. 238-243, 2018.
- [3] A. Singh, and C. Singh, "Analysis of Security Threats and Protocols for LTE Networks," *International Conference on Energy, Communication, Data Analytics and Soft Computing*, pp. 1660-1663, 2017.
- [4] L. Zhang, J. Bai, and P. Jiang, "Research on Key Management Scheme of X2 Handover Protocol in LTE-R," *IEEE 5th International Conference on Computer and Communications*, pp. 1479-1483, 2019.
- [5] Q. Xiao, W. Zhou, B. Cui, and L. Li, "An Enhancement for Key Management in LTE/SAE X2 Handover Based on Ciphering Key Parameters," *Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 256-261, 2014.

- [6] Q. Kong, R. Lu, S. Chen, and H. Zhu, "Achieve Secure Handover Session Key Management via Mobile Relay in LTE-advanced Networks," in *IEEE Internet of Things Journal*, Vol. 4, No. 1, pp. 29-39, 2017.
- [7] C. Y. Liu, F. Y. Leu, I. You, A. Castiglione, and F. Palmieri, "The Untrusted Handover Security of the S-PMIPv6 on LTE-A," *IEEE Conference on Computer Communications Workshops*, pp. 161-166, 2015.
- [8] K. A. Alezabi, F. Hashim, S. J. Hashim, and B. M. Ali, "An Efficient Authentication and Key Agreement Protocol for 4G (LTE) Networks," *IEEE Region 10 Symposium*, pp. 502-507, 2014.
- [9] G. Singh, and D. Shrimankar, "Dynamic Group Based Efficient Access Authentication and Key Agreement Protocol for MTC in LTE-A Networks," *Wireless Personal Communications*, pp. 829-856, 2018.
- [10] B. Parne, S. Gupta, and N. Chaudhari, "PSE-AKA: Performance and Security Enhanced Authentication Key Agreement Protocol for IoT Enabled LTE/LTE-A Networks," *Peer-to-Peer Networking and Applications*, Vol. 12, pp. 1156-1177, 2019.
- [11] G. Singh, and D. Shrimankar, "Security Analysis of LTE/SAE Networks with the Possibilities of Tampering E-UTRAN on NS3," *International Conference on Computing, Communication and Networking Technologies*, pp. 1-7, 2017.

- [12] M. A. Mobarhan, and M. Salamah, "REPS-AKA3: A Secure Authentication and Re-authentication Protocol for LTE Networks," *Journal of Network and Computer Applications*, Vol. 201, 2022.
- [13] S. Jiang, "Medium Access Control (MAC)," pp. 51-74, 2018.
- [14] A. Bohák, L. Buttyán, and L. Dóra, "An Authentication Scheme for Fast Handover between WiFi Access Points," *3rd International ICST Conference on Wireless Internet* , 2010.
- [15] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM SIGCOMM Computer Communication Review*, Vol. 33, pp. 93-102, 2003.
- [16] C. Zhang, R. Liu, H. Pin-Han, and A. Chen, "A Location Privacy Preserving Authentication Scheme in Vehicular Networks," *IEEE Wireless Communications and Networking Conference*, pp. 2543-2548, 2008.
- [17] H. Roh, and S. Jung, "RSA-based Proxy Signature for Media Independent Handover," *Journal of Measurement Science and Instrumentation*, pp. 349-353, 2010.
- [18] L. Tai, "Key Refresh During Cell Handover in LTE Featuring One-Hop Forward Security," *Computer Systems and Applications*, 2011.

- [19] S. Verma, Y. Kawamoto, and N. Kato, "A Network-Aware Internet-Wide Scan for Security Maximization of IPv6-Enabled WLAN IoT Devices," in *IEEE Internet of Things Journal*, Vol. 8, No. 10, pp. 8411-8422, 2021.
- [20] R. Gu, X. Zhang, L. Yu, and J. Zhang, "Enhancing Security and Scalability in Software Defined LTE Core Networks," *17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 837-842, 2018.
- [21] Q. T. Vien, T. A. Le, X. S. Yang, and T. Q. Duong, "On the Handover Security Key Update and Residence Management in LTE Networks," *IEEE Wireless Communications and Networking Conference*, pp. 1-6, 2017.
- [22] J. Cao, H. Li, M. Ma, Y. Zhang, and C. Lai, "A Simple and Robust Handover Authentication between HeNB and eNB in LTE Networks," *Computer Networks*, pp. 2119-2131, 2012.
- [23] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J. P. Seifert, "Practical Attacks against Privacy and Availability in 4G/LTE Mobile Communication Systems," *Network and Distributed System Security Symposium*, 2016.
- [24] F. Degefa, D. Lee, S. Kim, Y. Choi, and D. Won, "Performance and Security Enhanced Authentication and Key agreement protocol for SAE/LTE network," *Computer Networks*, Vol. 94, pp. 145-163, 2016.

- [25] K. Hamandi, J. Bou abdo, I. Elhajj, A. Kayssi, and A. Chehab, "A Privacy-Enhanced Computationally-Efficient and Comprehensive LTE-AKA," *Computer Communications*, Vol. 98, pp. 20-30, 2017.
- [26] M. A. Abdrabou, A. D. E. Elbayoumy, and E. A. El-Wanis, "LTE Authentication Protocol (EPS-AKA) Weaknesses Solution," *IEEE Seventh International Conference on Intelligent Computing and Information Systems*, pp. 434-441, 2015.
- [27] O. E. Ekene, R. Ruhl, and P. Zavorsky, "Enhanced User Security and Privacy Protection in 4G LTE Network," *IEEE 40th Annual Computer Software and Applications Conference*, pp. 443-448, 2016.
- [28] S. Abdolnezhad, and A. Sikora, "A Lightweight Mutual Authentication Protocol Based on Physical Unclonable Functions," *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 161-164, 2022.
- [29] N. Saxena, S. Grijalva, and S. Chaudhari, "Authentication Protocol for an IoT-enabled LTE Network," *ACM Transactions on Internet Technology*, Vol. 16, No. 25, pp. 1-20, 2016.
- [30] I. Elbouabidi, F. Zarai, M. Obaidat, and L. Kamoun, "An Efficient Design and Validation Technique for Secure Handover between 3GPP LTE and WLANs Systems," *Journal of Systems and Software*, Vol. 91, pp. 163-173, 2014.

- [31] J. Cao, M. Ma, H. Li, Y. Zhang, and Z. Luo, "A Survey on Security Aspects for LTE and LTE-A Networks," in *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, pp. 283-302, 2014.
- [32] C. K. Han, and H. K. Choi, "Security Analysis of Handover Key Management in 4G LTE/SAE Networks," in *IEEE Transactions on Mobile Computing*, Vol. 13, No. 2, pp. 457-468, 2014.
- [33] P. Agarwal, D. E. Thomas, and A. Kumar, "Security Analysis of LTE/SAE Networks under De-synchronization Attack for Hyper-erlang Distributed Residence Time," in *IEEE Communications Letters*, Vol. 21, No. 5, pp. 1055-1058, 2017.
- [34] P. K. Reddy, and B. R. Chandavarkar, "Mitigation of Desynchronization Attack during Inter-eNodeB Handover Key Management in LTE," *Eighth International Conference on Contemporary Computing*, pp. 561-566, 2015.
- [35] M. Abdeljebbar, and R. Kouch, "Fast Authentication during Handover in 4G LTE/SAE Networks," *IERI Procedia*, Vol. 10, pp. 11-18, 2014.
- [36] Z. Haddad, "Secure and Efficient Uniform Handover Scheme for LTE-A Networks," *IEEE Wireless Communications and Networking Conference*, pp. 457-468, 2016.
- [37] B. Sridevi, D. Mohan, and R. Neelaveni, "Secured Handover Key Management among LTE Entities Using Device Certification," *3rd*

International Conference on Eco-friendly Computing and Communication Systems, pp. 155-160, 2014.

- [38] M. Ouaisa, A. Rhattoy, and I. Chana, "New Security Level of Authentication and Key Agreement Protocol for the IoT on LTE Mobile Networks," *6th International Conference on Wireless Networks and Mobile Communications*, pp. 1-6, 2018.
- [39] A. B. Seyi, F. Jafaar, and R. Ruhl, "Securing the Authentication Process of LTE Base Stations," *International Conference on Electrical, Communication, and Computer Engineering*, pp. 1-6, 2020.
- [40] J. Huang, J. Liu, Y. Yan, G. Zhao, Y. Li, and W. Luo, "The Enhanced EAP Authentication Protocol for Devices in Power LTE Private Network," *3rd World Symposium on Communication Engineering*, pp. 49-53, 2020.
- [41] S. Garima, and D. Shrimankar, "A Privacy-Preserving Authentication Protocol with Secure Handovers for the LTE/LTE-A Networks," *Sādhana*, Vol. 43, pp. 1-18, 2018.
- [42] B. L. Parne, S. Gupta, and N. S. Chaudhari, "SEGB: Security Enhanced Group Based AKA Protocol for M2M Communication in an IoT Enabled LTE/LTE-A Network," in *IEEE Access*, Vol. 6, pp. 3668-3684, 2018.

- [43] M. Abdeljebbar, and R. Kouch, "Security Improvements of EPS-AKA Protocol," *International Journal of Network Security*, Vol. 20, No. 4, pp. 636-644, 2018.
- [44] J. Bou abdo, and J. Demerjian, "Security V/S QoS for LTE Authentication and Key Agreement Protocol," *International Journal of Network Security ITS Applications*, pp. 71-82, 2012.
- [45] K. Hamandi, I. Sarji, A. Chehab, I. H. Elhajj, and A. Kayssi, "Privacy Enhanced and computationally Efficient HSK-AKA LTE Scheme," *27th International Conference on Advanced Information Networking and Applications Workshops*, pp. 929-934, 2013.
- [46] C. Zemaio, Z. Junge, and H. Biyi, "Optimizing PKI for 3GPP Authentication and Key Agreement," *Fourth International Conference on Multimedia Information Networking and Security*, pp. 79-82, 2012.
- [47] H. Yang, V. Oleshchuk, and A. Prinz, "Verifying Group Authentication Protocols by Scyther," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl*, pp. 3-19, 2016.
- [48] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. Comput. Syst.* 8, pp. 18-36, doi:10.1145/74850.74852.

- [49] J. Cao, H. Li, and M. Ma, "GAHAP: A Group-based Anonymity Handover Authentication Protocol for MTC in LTE-A Networks," *IEEE International Conference on Communications*, pp. 3020-3025, 2015.
- [50] J. Cao, H. Li, M. Ma, and F. Li, "UGHA: Uniform Group-based Handover Authentication for MTC within E-UTRAN in LTE-A Networks," *IEEE International Conference on Communications*, pp. 7246-7251, 2015.
- [51] Y. W. Chen, J. T. Wang, K. H. Chi, and C. C. Tseng, "Group-based Authentication and Key Agreement," *Wireless Personal Communications*, pp. 965-979, 2012.
- [52] R. Jiang, C. Lai, J. Luo, X. Wang, and H. Wang, "EAP-based Group Authentication and Key Agreement Protocol for Machine-type Communications," *International Journal of Distributed Sensor Networks*, Vol. 9, 2013.
- [53] J. Cao, M. Ma, and H. Li, "GBAAM: Group-based Access Authentication for MTC in LTE Networks," *Security and Communication Networks*, Vol. 8, 2015.
- [54] D. Choi, H. K. Choi, and S. Y. Lee, "A Group-based Security Protocol for Machine-type Communications in LTE-advanced," *IEEE Conference on Computer Communications Workshops*, pp. 161-162, 2014.

- [55] M. N. Khan, A. Rao, and S. Camtepe, "Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey," in *IEEE Internet of Things Journal*, Vol. 8, No. 6, pp. 4132-4156, 2021.
- [56] A. Fu, J. Song, S. Li, G. Zhang, and Y. Zhang, "A Privacy-Preserving Group Authentication Protocol for Machine-Type Communication in LTE/LTE-A Networks," *Security and Communication Networks*, Vol. 9, pp. 2002-2014, 2016.
- [57] J. Yao, T. Wang, M. Chen, L. Wang, and G. Chen, "GBS-AKA: Group-based Secure Authentication and Key Agreement for M2M in 4G Network," *International Conference on Cloud Computing Research and Innovations*, pp. 42-48, 2016.
- [58] C. Lai, H. Li, R. Lu, and X. Shen, "SE-AKA: A Secure and Efficient Group Authentication and Key Agreement Protocol for LTE Networks," *Computer Networks*, Vol. 57, pp. 3492-3510, 2013.
- [59] C. Lai, H. Li, X. Li, and J. Cao, "A Novel Group Access Authentication and Key Agreement Protocol for Machine-type Communication," *Transactions on Emerging Telecommunications Technologies* Volume, Vol. 26, pp. 414-431, 2015.

- [60] J. Li, M. Wen, and T. Zhang, "Group-based Authentication and Key Agreement with Dynamic Policy Updating for MTC in LTE-A Networks," *IEEE Internet of Things Journal*, Vol. 3, No. 3, pp. 408-417, 2016.
- [61] M. Abdeljebbar, and R. Elkouch, "Security Analysis of LTE/SAE Networks over E-UTRAN," *International Conference on Information Technology for Organizations Development*, pp. 1-5, 2016.
- [62] G. M. Køien, "Mutual Entity Authentication for LTE," *7th International Wireless Communications and Mobile Computing Conference*, pp. 689-694, 2011.
- [63] X. Li, and Y. Wang, "Security Enhanced Authentication and Key Agreement Protocol for LTE/SAE Network," *7th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, 2011.
- [64] M. Purkhiabani, and A. Salahi, "Enhanced Authentication and key Agreement Procedure of next Generation Evolved Mobile Networks," *IEEE 3rd International Conference on Communication Software and Networks*, pp. 557-563, 2011.
- [65] H. Choudhury, B. Roychoudhury, and D. K. Saikia, "Enhancing User Identity Privacy in LTE," *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 949-957, 2012.

- [66] C. Lai, R. Lu, H. Li, D. Zheng, and X. Shen, "Secure Machine-type Communications in LTE Networks," *International Journal of Wireless Communications and Mobile Computing*, Vol. 16, No. 2, pp. 1495-1509, 2015.
- [67] X. Cheng, M. Ma, X. Huang, and H. Bao, "A Cross-domain Group Authentication Scheme for LTE-A based Vehicular Network," *9th IEEE International Conference on Communication Software and Networks*, pp. 595-599, 2017.
- [68] S. Gupta, B. Parne, and N. Chaudhari, "DGBES: Dynamic Group Based Efficient and Secure Authentication and Key Agreement Protocol for MTC in LTE/LTE-A Networks," *Wireless Personal Communications: An International Journal* Volume, Vol. 98, pp. 2867-2899, 2018.
- [69] M. Modiri, J. Mohajeri, and M. Salmasizadeh, "GSL-AKA: Group-based Secure Lightweight Authentication and key Agreement Protocol for M2M Communication," *9th International Symposium on Telecommunications*, pp. 275-280, 2018.
- [70] Cao, H. Li, M. Ma, and F. Li, "UPPGHA: Uniform Privacy Preservation Group Handover Authentication Mechanism for MTC in LTE-A Networks," *Journal of Security & Communication Network*, pp. 1-16, 2018.
- [71] C. Lai, R. Lu, D. Zheng, H. Li, and X. S. Shen, "GLARM: Group-based Lightweight Authentication Scheme for Resource-constrained Machine to

Machine Communications," *International Journal of Computer Network*, Vol. 99, No. 6, pp. 66-81, 2016.

- [72] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security," in *IEEE Internet of Things Journal*, Vol. 7, No. 10, pp. 10250-10276, 2020.
- [73] B. Gupta, N. Nedjah, P. Vijayakumar, A. Ahmed, and X. Chang, "Machine LEarning and Smart Card BAsed Two-Factor Authentication Scheme for Preserving Anonymity in Telecare Medical Information System (TMIS)," *Neural Computing and Applications*, 2021.
- [74] M. Purkhiabani, and A. Salahi, "Enhanced Authentication and Key Agreement Procedure of Next Generation 3GPP Mobile Networks," *IEEE 3rd International Conference on Communication Software and Networks*, pp. 557-563, 2011.
- [75] J. B. Abdo, J. Demerjian, K. Ahmad, H. Chaouchi, and G. Pujolle, "EPS Mutual Authentication and Crypt-analyzing SPAKA," *International Conference on Computing, Management and Telecommunications*, pp. 303-308, 2013.

- [76] J. B. Abdo, J. Demerjian, H. Chaouchi, and G. Pujolle, "EC-AKA2 a Revolutionary AKA Protocol," *International Conference on Computer Applications Technology*, pp. 1-6, 2013.
- [77] D. Shin, K. Yun, J. Kim, P. Virgil, J. N. Kim, and I. You, "A Security Protocol for Route Optimization in DMM-Based Smart Home IoT Networks," *IEEE Access*, Vol. 7, 2019.
- [78] W. Zhang, A. Elgendy ,M. Hammad, A. M. Iliyasu , D. Xiaojiang, M. Guizani, A. Ahmed, and A. El-Latif, "Secure and Optimized Load Balancing for Multitier IoT and Edge-cloud Computing Systems," in *IEEE Internet of Things Journal*, Vol. 8, No. 10, pp. 8119-8132, 2021.
- [79] Y. Liu, J. Peng, J. Kang, M. Abdullah, D. Niyato, and A. Abd El-Latif, "A Secure Federated Learning Framework for 5G Networks," *IEEE Wireless Communications*, Vol. 27, No. 4, pp. 24-31, 2020.
- [80] N. Qachri, and J. M. Dricot, "On the Security of WLAN Access Points Integrated in 4G/LTE Architectures," *19th IEEE Workshop on Local & Metropolitan Area Networks*, pp. 1-6, 2013.
- [81] R. Rajavelsamy, D. Das, and M. Choudhary, "Privacy Protection and Mitigation of Unauthorized Tracking in 3GPP-WiFi Interworking Networks," *IEEE Wireless Communications and Networking Conference*, pp. 1-6, 2018.

- [82] K. Alezabi, F. Hashim, S. Hashim, B. Mohd Ali, and A. Jamalipour, "On the Authentication and Re-authentication Protocols in LTE-WLAN Interworking Architecture," *Transactions on Emerging Telecommunications Technologies*, Vol. 28, No. 4, 2016.
- [83] L. Cao, Z. Yu, Y. Liu, and S. Cao, "An Identity Privacy Protection Scheme in LTE-WLAN Heterogeneous Converged Network," *IEEE Access*, Vol. 9, pp. 78356-78367, 2021.
- [84] A. Hassanein, A. Hafez, and A. Gaafar, "New Authentication and Key Agreement Protocol for LTE-WLAN Interworking," *International Journal of Computer Applications*, Vol. 61, pp. 20-24, 2013.
- [85] IEEE, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications," 2009.
- [86] D. Kakadia, and A. Gilgur, "Evolved Universal Terrestrial Radio Access Network (EUTRAN)," *Network Performance and Fault Analytics for LTE Wireless Service Providers*, pp. 61-81, 2017.
- [87] X. Wang, X. She, L. Bai, Y. Qing, and F. Jiang, "A Novel Anonymous Authentication Scheme Based on Edge Computing in Internet of Vehicles," *Computers, Materials and Continua*, Vol. 67, pp. 3349-3361, 2021.
- [88] O. Abiona, A. Oluwaranti, A. Oluwatope, S. Bello, C. Onime, M. Sanni, and L. Kehinde, "Wireless Network Security: The Mobile Agent Approach,"

International Journal of Communications, Network and System Sciences,
Vol. 06, pp. 443-450, 2013.

- [89] K. A. Alezabi, F. Hashim, S. J. Hashim, and B. M. Ali, "A New Tunnelled EAP Based Authentication Method for WiMAX Networks," *IEEE 11th Malaysia International Conference on Communications*, pp. 412-417, 2013.
- [90] T. Feng, H. Chen, and J. f. Ma, "Secure Re-authentication Scheme for 3G-WLAN Integrating Network Based on Protocol Composition Logic," *International Conference on Computer Science and Service System*, pp. 800-805, 2012.
- [91] Y. El Hajjaji El Idrissi, N. Zahid, and M. Jedra, "A New Fast Re-authentication Method for the 3G-WLAN Interworking based on EAP-AKA," *ICT*, pp. 1-5, 2013.
- [92] S.-S. Shen, S.-H. Lin, and C. Hui, "Fast Handover Preauthentication Protocol in 3GPP-WLAN Heterogeneous Mobile Networks," *International Journal of Communications, Network and System Sciences*, Vol. 07, pp. 101-113, 2014.
- [93] Y. Idrissi, N. Zahid, and M. Jedra, "A New Authentication Method for Vertical and Horizontal HAndover in 3G-WLAN Interworking Architecture," *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 7, pp. 329-342, 2014.

- [94] 3GPP, "Access to the 3GPP Evolved Packet Core (EPC) via Non-3GPP Access Networks Stage 3 (Release 9)," 3GPP TS 24.302 V9.4.0, 2010.
- [95] 3GPP, "Numbering, Addressing and Identification (Release 10)," TS 23.003 v10.5.0, 2012.
- [96] H. Choi, O. Song, and D. H. Cho, "Seamless Handoff Scheme Based on Pre-Registration and Pre-authentication for UMTSWLAN Interworking," *Wireless Personal Communications volume*, Vol. 41, pp. 345-364, 2007.
- [97] A. A. Al Shidhani, and V. C. M. Leung, "Fast and Secure Reauthentications for 3GPP Subscribers during WiMAX-WLAN Handovers," in *IEEE Transactions on Dependable and Secure Computing*, Vol. 8, No. 5, pp. 699-713, 2011.
- [98] R. H. Hwang, C. F. Huang, H. W. Lin, and J. J. Wu, "Uplink Access Control for Machine-type communications in LTE-A Networks," *Personal and Ubiquitous Computing*, pp. 851-862, 2016.
- [99] Z. Wang, P. Sun, N. Luo, and B. Guo, "A Three-Party Mutual Authentication Protocol for Wearable IoT Health Monitoring System," *IEEE International Conference on Smart Internet of Things*, pp. 344-347, 2021.

- [100] J. Cao, M. Ma, and H. li, "A Group-based Authentication and Key Agreement for MTC in LTE Networks," *IEEE Global Communications Conference*, pp. 1017-1022, 2012.
- [101] 3GPP, "System Improvements for Machine-type Communications (Release 11)," Technical Specification (TS) 23.888, 3rd Generation Partnership Project (3GPP), version 1.4.0 (08 2011).
- [102] C. Lai, H. Li, R. Lu, Rong Jiang, and X. Shen, "LGTH: A Lightweight Group Authentication Protocol for Machine-type Communication in LTE Networks," *IEEE Global Communications Conference*, pp. 832-837, 2013.
- [103] C. Lai, R. Lu, D. Zheng, H. Li, X. Shen, "Toward Secure Large-scale Machine-to-Machine Communications in 3GPP Networks: Challenges and Solutions," *IEEE Communications Magazine*, Vol. 53, pp. 12-19. 2015.
- [104] S. B. Sarvaiya, and D. N. Satange, "Security in IP-Based IoT Node and Device Authentication," *IEEE International Conference on Blockchain and Distributed Systems Security*, pp. 1-5, 2022.
- [105] K. K. Nair, and H. D. Nair, "Security Considerations in the Internet of Things Protocol Stack," *International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems*, pp. 1-6, 2021.

- [106] M. Mobarhan, and M. Salamah, "REPS-AKA5: A Robust Group-based Authentication Protocol for IoT Applications in LTE System," *Internet of Things Journal*, Vol. 22, 2023.
- [107] X. Zhang, A. Kunz, and S. Schröder, "Overview of 5G security in 3GPP," *IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 181-186, 2017.
- [108] A. Dutta, and E. Hammad, "5G Security Challenges and Opportunities: A System Approach," *IEEE 3rd 5G World Forum (5GWF)*, pp. 109-104, 2020.
- [109] Y. Guo, and Y. Zhang, "Study on Core Network Security Enhancement Strategies in 5G Private Networks," *IEEE 21st International Conference on Communication Technology (ICCT)*, pp. 887-891, 2021.
- [110] H. Kholidy, A. Karam, J. Sidoran, and M. Rahman, "5G Core Security in Edge Networks: A Vulnerability Assessment Approach," *IEEE Symposium on Computers and Communications (ISCC)*, pp. 1-6, 2021.

APPENDICES

Appendix 1: AVISPA codes

%% PROTOCOL: REPS-AKA1

```
Role role_U(UE,HH,VM:agent,IMSI,MVP:text,H1,H2,H3,H4,H5,H12:hash_func,H10,H11,AUTHFunc,KDF,F1,F2,F3,F4 ,F5:hash_func, OSK:symmetric_key,Kpm,Kph,Kpu: public_key, SND_VM,RCV_VM:channel(dy))played_by UE def=local State:nat,RK,Ru:text,SQN,RAND, AUTH,RNK,MA,AMF,HSNID,HMVP,NNK,HIMSI,HRu,UIH, XRNK,XMA,XMAC ,RES,KASME,CK,IK,AK,HRES,Kum:text const ue_mme_IMSI, ue_mme_HIMSI, ue_mme_HRu,ue_mme_Kum, ue_mme_HMVP,ue_mme_RES,ue_hhs_UIH,ue_hhs_HMVP,ue_hhs_HIMSI,mme_ue_auth_HRES,ue_hss_auth_AUTH,ue_mme_auth_HMVP,mme_ue_msg_auth_NNK,ue_mmsg_auth_NNK:protocol_id init State := 0 transition 1. State=0 /RCV_VM(start)=> State:=1/\ RK':=new() /\ NNK' :=xor(RK', OSK) /\HIMSI':=H1(IMSI)/\Ru':=new() /\HRu':=H2(Ru') /\HMVP':=H3(MVP)/\UIH':=H4(xor(IMSI,OSK))/\Kum':=H5(xor(IMSI,NNK'))/\SND_VM'({HIMSI'.HRu'.HMVP'.Kum'. {NNK'. inv(Kpu)}_Kpm.{HIMSI'.HMVP.UIH'}_Kph)/\secret(IMSI,ue_mme_IMSI,{UE,VM})/\secret(HIMSI ,ue_mme_HIMSI, {UE,VM})/\ secret(HRu ,ue_mme_HRu,{UE,VM})/\ secret(Kum,ue_mme_Kum,{UE,VM}) /\ secret(HMVP ,ue_mme_HMVP, {UE, VM}) /\ secret(UIH,ue_hhs_UIH,{UE,HH}) /\ secret(HMVP ,ue_hhs_HMVP,{UE,HH}) /\secret(HIMSI,ue_hhs_HIMSI,{UE,HH}) /\witness(UE,VM,mme_ue_msg_auth_NNK,NNK') /\witness(UE,HH,hss_ue_msg_auth_NNK,NNK ')\ State=1/RCV_VM'({SQN'.RAND'.AUTH'.HSNID'}_Kum'.{RNK'.MA'.AMF'}_Kpu') => State':=2 /\ HMVP':= H3(MVP) /\ UIH' := H4(xor(IMSI,OSK)) /\ XRNK':= H10(UIH') /\XMA':=H11(XRNK'.HSNID') /\XMAC':=F1(XRNK'.RAND'.AMF',SQN') /\RES':=F2(XRNK'.RAND') /\CK':=F3(XRNK'.RAND') /\ IK':=F4(XRNK'.RAND') /\AK':=F5(XRNK'.RAND') /\KASME':=KDF(xor(SQN',AK'),HSNID'.CK'.IK') /\AUTH':=AUTHFunc(xor(SQN',AK'),AMF'.XMAC') /\HRES':=H12(RES') /\ SND_VM'({HRES'}_Kum') /\secret(HRES,ue_mme_RES,{UE,VM}) /\witness(UE,VM,mme_ue_auth_HRES,HRES') /\request(UE,HH,ue_hss_auth_AUTH,AUTH') /\request (UE,VM,ue_mme_auth_HMVP,HMVP)end role role_VM(VM,HH,UE:agent,HVP,MVP,SNID:text,H6,H7,H8,H9:hash_func,Kpm,Kph,Kpu: public_key, SND_HH,RCV_HH,SND_UE,RCV_UE:channel(dy)) played_by VM def= local State:nat,HMVP,HIMSI, HRu,KASME, SQN, RAND,RES,XRES,NNK,UIH, SIND,AUTH,HRES,RNK,MA,AMF,HSNID, HHVP, HNID,Kum,Kmh,XHMVP:text, X: {text.text.text}_public_key const mme_hss_kmh, mme_hss_HHVP, mme_hss_HSNID, mme_hss_UIH, mme_hss_HVP, mme_hss_HIMSI, mme_ue_SQN, mme_ue_RAND, mme_ue_AUTH, mme_ue_MA, mme_ue_RNK, mme_ue_AMF, mme_ue_auth_HRES, ue_mme_auth_HMVP, mme_hss_auth_HNID, mme_ue_msg_auth_NNK : protocol_id init State := 0 transition 1. State=0 /\ RCV_UE'({HIMSI'.HRu'.HMVP'.Kum'.{NNK'}_inv(Kpu)}_Kpm.{HIMSI'.HMVP.UIH'}_Kph.X') => State':=1 /\ HSNID':= H6(SNID) /\Kmh':= H7(xor(SNID, HIMSI')) /\ HHVP':= H8(HVP) /\ HNID':= H9(xor(HVP, MVP)) /\SND_HH'(X':{Kmh'.HMVP'. HSNID'.HNID'.{NNK'}_inv(Kpu)}_Kph /\ secret(Kum', mme_hss_kmh, {VM, HH}) /\secret(HHVP',mme_hss_HHVP,{VM,HH}) /\secret(HSNID',mme_hss_HSNID,{VM,HH}) /\request(VM,UE,mme_ue_msg_auth_NNK,NNK')\ State=1/RCV_H H'({HNID'.SQN'.RAND'. AUTH'. KASME'. HRES'. XHMVP'}_Kmh'.{RNK'.MA'.AMF'}_Kpu') => State':=2 /\ HMVP':= XHMVP' /\ HNID':= H9(xor(HVP, MVP)) /\ SND_UE'({SQN'.RAND'. AUTH'.HMVP'}_Kum. {RNK'.MA'.AMF'}_Kpu') /\ secret(SQN',mme_ue_SQN, {VM,UE}) /\secret(RAND',mme_ue_RAND, {VM,UE}) /\ secret(AUTH',mme_ue_AUTH, {VM,UE}) /\ secret(MA',mme_ue_MA, {VM,UE}) /\ secret(RNK',mme_ue_RNK, {VM,UE}) /\ secret(AMF',mme_ue_AMF, {VM, UE}) /\ witness (VM, UE, ue_mme_auth_HMVP, HMVP) /\ request (VM, HH, mme_auth_HNID, HNID') 5. State=2 /\ RCV_UE'({HRES'}_Kum') => State':=3 /\ request (VM, UE, mme_ue_auth_HRES, HRES') end role role_HH(HH, VM, UE:agent,IMSI, HVP, MVP:text, H1, H3, H8, H4, H9, H10, H11, H12 :hash_func, F1, F2, F3, F4, F5, KDF, AUTHFunc :hash_func, OSK:symmetric_key,Kpm,Kph,Kpu: public_key,SND_VM,RCV_VM:channel(dy)) played_by HH def=local State:nat, HNID, HIMSI, HMVP, HHVP, UIH, HSNID, AMF, Kmh, XHIMSI, XHMVP, XHHVP, XUIH, RNK, MA, SQN, MAC, XRES, CK, IK, AK, KASME, AUTH,NNK,HRES,RAND,SNID:textconst hss_mme_KASME,hss_mme_XRES, hss_mme_AUTH,hss_mme_RAND,hss_mme_SQN,hss_mme_HNID,hss_mme_RNK, hss_mme_MA, hss_mme_AMF, ue_hss_auth_AUTH, hss_ue_msg_auth_NNK:protocol_id init State:=0transition2.State=0 /\ RCV_VM'({HIMSI'.HMVP. UIH'}_Kph.{Kmh'.HHVP'.HSNID'.HNID'.{NNK'}_inv(Kpu)}_Kph') =>State':=1 /\XHIMSI':= H1(IMSI) /\XHHVP':=H8(HVP) /\XHMVP':=H3 (MVP') /\ HNID' :=H9(xor (HVP,MVP)) /\XUIH':=H4(xor(IMSI,OSK)) /\RNK':=H10(XUIH') /\MA':=H11(RNK'.HSNID') /\RAND':=new() /\SQN':=new() /\ AMF':=new() /\MAC':=F1(RNK'.RAND'. AMF',SQN') /\XRES':=F2(RNK'. RAND') /\ HRES':=H12(XRES') /\ CK':=F3(RNK'.RAND') /\ IK':=F4(RNK'.RAND') /\ AK':=F5(RNK'.RAND') /\KASME':=KDF (xor(SQN',AK'),HSNID'.CK'.IK') /\AUTH':=AUTHFunc(xor(SQN',AK'),AMF'.MAC') /\SND_VM'({HNID'.SQN'.RAND'.AUTH'.KASME'.HRES'.XHMVP'}_Kmh'.{RN K'. MA'.AMF'}_Kpu) /\ secret(HNID',hss_mme_HNID, {HH, VM}) /\ secret(SQN',hss_mme_SQN, {HH, VM}) /\ secret(RAND',hss_mme_RAND, {HH, VM}) /\ secret(AUTH',hss_mme_AUTH, {HH, VM}) /\ secret(XRES',hss_mme_XRES, {HH, VM}) /\ secret(KASME',hss_mme_KASME, {HH, VM}) /\ secret(RNK' ,hss_mme_RNK, {HH, VM}) /\ secret(MA',hss_mme_MA, {HH, VM}) /\ secret(AMF',hss_mme_AMF, {HH, VM}) /\ witness (HH, UE, ue_hss_auth_AUTH, AUTH') /\ witness (HH, VM, mme_hss_auth_HNID, HNID') /\ request (HH, UE, hss_ue_msg_auth_NNK,NNK') end role role_session1(HH,VM,UE:agent IMIS,RK, Ru, HSNID, NNK, UIH, XRNK, XMA, XMAC, CK, IK, AK, HRES : text, HVP,MVP, HHVP, HNID, HMVP,HIMSI, HRu, KASME, RES : text, AMF, XHIMSI, XHMVP, XHHVP, XUIH, RNK, MA, RAND, SQN, MAC, XRES, AUTH, SNID,Kmh: text, H1, H2, H3, H4, H5, H6, H7, H8, H9, H10, H11, H12, F1 ,F2 ,F3 ,F4, F5, KDF, AUTHFunc: hash_func, Kpm,Kph,Kpu: public_key, OSK:symmetric_key) def=local SND_VM1, RCV_VM1, SND_HH2, RCV_HH2, SND_UE2, RCV_VM3, RCV_VM3 : channel(dy) composition role_U(UE, VM, HH, IMSI,MVP, H1, H2, H3, H4, H5, H10, H11, H12, F1 ,F2 ,F3 ,F4, F5, KDF, AUTHFunc,OSK, Kpm, Kph, Kpu,SND_VM1, RCV_VM1) /\ role_VM(VM, HH, UE, HVP,MVP, SNID, H6, H7, H8, H9,Kpm, Kph, Kpu, SND_HH2, RCV_HH2, SND_UE2, RCV_UE2) /\ role_HH(HH, VM, UE, IMSI, HVP, MVP, H1, H3, H8, H4, H9, H10, H11, H12, F1, F2, F3, F4, F5, KDF, AUTHFunc, OSK, Kpm, Kph, Kpu, SND_VM3, RCV_VM3) end role role_environment(def=constmme:agent,hss:agent,ue:agent, imsi,rk, ru, hsnid, nnk, uih, xrnk, xma, xmac, ck, ik, ak, hres : text, hvp,mvp, hhvp, hnid, hmv,himsi, hr, kum, kasme,res : text, amf, xhimsi, xhmvp, xhhvp, xuih, rk, ma, rand, sqn, mac, xres, auth, snid,kmh: text, h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, f1 ,f2 ,f3 ,f4, f5, kdf, authfunc:hash_func, kpm,kph,kpu:public_key, osk:symmetric_key, ue_mme_IMSI, ue_mme_HIMSI, ue_mme_HRu, ue_mme_Kum,ue_mme_HMVP, ue_mme_RES, ue_hhs_UIH, ue_hhs_HMVP, ue_hhs_HIMSI : protocol_id, mme_ue_auth_HRES, ue_hss_auth_AUTH, ue_mme_auth_HMVP, mme_ue_msg_auth_NNK, hss_ue_msg_auth_NNK: protocol_id, mme_hss_kmh, mme_hss_HHVP, mme_hss_HSNID, mme_hss_UIH, mme_hss_HVP, mme_hss_HIMSI : protocol_id, mme_ue_SQN, mme_ue_RAND, mme_ue_AUTH, mme_ue_RNK, mme_ue_MA, mme_ue_AMF: protocol_id, mme_hss_auth_HNID, mme_ue_msg_auth_NNK : protocol_id, hss_mme_KASME, hss_mme_XRES, hss_mme_AUTH, hss_mme_RAND,hss_mme_SQN,hss_mme_HNID : protocol_id, hss_mme_RNK, hss_mme_MA, hss_mme_AMF : protocol_id, intruder_knowledge = {h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, f1 ,f2 ,f3 ,f4, f5, kdf, authfunc, kpm,kph,kpu} composition session1(mme,hss,ue, imsi,rk, ru, hsnid, nnk, uih, xrnk, xma, xmac, ck, ik, ak, hres, hvp,mvp, hhvp,hnid, hmv,himsi, hr, kasme,res, amf, xhimsi, xhmvp, xhhvp, xuih, rnk, ma, rand, sqn, mac, xres, auth, snid,kmh,h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, f1 ,f2 ,f3 ,f4, f5, kdf, authfunc,kpm,kph,kpu,osk) end role goalauthentication_on mme_ue_auth_HRES authentication_on ue_hss_auth_AUTH authentication_on ue_mme_auth_HMVP authentication_on mme_ue_msg_auth_NNK authentication_on hss_ue_msg_auth_NNK authentication_on mme_hss_auth_HNID secrecy_of ue_mme_IMSI secrecy_of ue_mme_HIMSI secrecy_of ue_mme_HRu secrecy_of ue_mme_Kum secrecy_of ue_mme_HMVP secrecy_of ue_mme_RES secrecy_of ue_hhs_UIH secrecy_of ue_hhs_HMVP secrecy_of ue_hhs_HIMSI secrecy_of mme_hss_kmh secrecy_of mme_hss_HHVP secrecy_of mme_hss_HSNID secrecy_of mme_hss_UIH secrecy_of mme_hss_HVP secrecy_of mme_hss_HIMSI secrecy_of mme_ue_SQN secrecy_of mme_ue_RAND secrecy_of mme_ue_AUTH secrecy_of mme_ue_MA secrecy_of mme_ue_RNK secrecy_of mme_ue_AMF secrecy_of hss_mme_KASME secrecy_of hss_mme_XRES secrecy_of hss_mme_AUTH secrecy_of hss_mme_AMF end goal environment()
```

%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.06s, visitedNodes: 2 nodes, depth: 1 plies

%%Statistics (ATSE mode): Analysed: 2 states, Reachable: 0 states, Translation: 0.02 seconds, Computation: 0.00 seconds

Explanation of statistics parameters:

OFMC mode:

parseTime: The time taken to parse the input file.

searchTime: The time taken to search for an attack in the model. In this case, it took 0.06 seconds to complete the search.

visitedNodes: The number of nodes visited during the search. In this case, it was 2 nodes.

depth: The maximum depth of the search. In this case, the search only went to a depth of 1 plie.

ATSE mode:

Analysed: The number of states analyzed by the tool.

Reachable: The number of states that were found to be reachable.

Translation: The time taken to translate the model into the internal representation used by the tool. In this case, it took 0.02 seconds to perform the translation.

Computation: The time taken to analyze the model. In this case, it took 0.00 seconds to perform the analysis.

NOTE1: Computation: 0.00 seconds. This can happen when the model is small and simple enough that the tool can easily determine that there are no security vulnerabilities in the protocol. In such cases, the tool can prove the security properties of the model without having to perform any complex computations.

NOTE2: for other modeled protocol, we have the same parameters and descriptions.

%% PROTOCOL: REPS-AKA2

```
role_U(UE,ENBS,ENBT,MME:agent,IMSI,T1,UECap,T1N,UECapN,EID1,E1ueAuth,KASME,ASIAD,ASEAD,ASUPEAD,NASCAD, NASIAD,KeNB, PE2ueAuth,PKRRRCint,PKRRRCenc,PKRRRCUPen, PKNASenc,PKNASint:text, H1, AuthFunc, NKGFunc, KDF, UKGFunc:hash_func, PKue,PKe2,PKe1,pKum: public_key, PFSKe2:symmetric_key, SND_ENBT,RCV_ENBT:channel(dy)) played_by UE def=local State:nat,HIMSI, E1ueAuthN,RNue : text, RNrNe2,RNue1,T4,NKG,KeNB,N,KRRRCint,KRRRCenc,KRRRCUPen,KASME,N,KNASenc,KNASint,E2ueAuth,FSKe2u:text, RNueN,E2ueAuthN, T2N,RNue2N, Xe2ueAuth,Xe1ueAuthN,UKue,E2ueAuthNN:text, KRRRCintN,KRRRCencN,KRRRCUPenN,KNASencN,KNASintN: text const ue_enbt_IMSI, ue_enbt_HIMSI,ue_enbt_UECap, ue_enbt_e1ueAuthN,ue_enbt_e2ueAuthN,ue_enbt_RNe2,ue_enbt_e2ueAuth,ue_enbt_auth_RNue, ue_enbt_msg_auth_FSKe2u,ue_enbt_e2ueAuth,ue_enbt_RNe2,ue_enbt_e2ueAuthN,ue_enbt_e2ueAuth,ue_enbt_auth_RNue: protocol_id init State := 0 transition 1. State=0 /\ RCV_ENBT(start)=> State:=1/\ RNue':=new() /\ HIMSI':= H1(IMSI)/\E1ueAuthN':= AuthFunc(xor (E1ueAuth, RNue')) /\ SND_ENBT'({HIMSI'.UECap. RNue'.EID1.PKue,{E1ueAuthN'}_inv(PKue))_PKe2 /\ secret(IMSI,ue_enbt_IMSI, {UE, ENBT}) /\ secret(HIMSI' ,ue_enbt_HIMSI, {UE, ENBT}) /\ secret(UECap ,ue_enbt_UECap, {UE, ENBT}) /\ secret(E1ueAuthN' ,ue_enbt_e1ueAuthN, {UE, ENBT}) /\ witness (UE,ENBT,
```

265

authentication_on enbt_ue_msg_auth_e1ueAuthNauthentication_on ue_enbt_auth_RNue authentication_on ue_enbt_msg_auth_FSKe2u authentication_on enbt_ue_msg_auth_e2ueAuthNauthentication_on enbt_ue_auth_RNe2 authentication_on enbt_ue_msg_auth_e1ueAuthNauthentication_on ue_enbt_msg_auth_e2ueAuthNauthentication_on enbt_ue_auth_RNe2 authentication_on enbt_ue_auth_RNueN authentication_on enbt_enbs_msg_auth_Xe1ueAuthN authentication_on enbt_enbs_auth_RNe2 authentication_on enbs_enbt_msg_auth_HIMSI authentication_on enbt_mvp authentication_on mme_enbt_msg_auth_MVP secrecy_of ue_enbt_IMSI_of ue_enbt_HIMSI secrecy_of ue_enbt_UECap secrecy_of ue_enbt_e1ueAuthN secrecy_of ue_enbt_RNe2 secrecy_of ue_enbt_e2ueAuth secrecy_of enbt_enbs_UECap secrecy_of enbt_enbs_RNe2 secrecy_of enbt_enbs_eID2 secrecy_of enbt_enbs_HIMSI secrecy_of enbt_enbs_e1ueAuthN secrecy_of enbt_ue_Xe1ueAuthN secrecy_of enbt_ue_RNe2 secrecy_of enbt_ue_RNe1 secrecy_of enbt_mme_UECap secrecy_of enbt_mme_RNe2N secrecy_of enbt_mme_UK secrecy_of enbt_mme_HIMSI secrecy_of enbt_mme_MVP secrecy_of enbt_ue_RNe2N secrecy_of enbt_ue_RNueN secrecy_of enbt_ue_Xe2ueAuth secrecy_of enbt_ue_RNue secrecy_of enbt_ue_FSKe2u secrecy_of enbs_enbt_HIMSI secrecy_of enbs_enbt_KeNBn secrecy_of enbs_enbt_Xe1ueAuthN secrecy_of enbs_mme_HIMSI secrecy_of enbs_mme_UECap secrecy_of enbs_mme_eID2 secrecy_of enbs_mme_NKG secrecy_of enbs_mme_MVP end goal environment()

%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.11s, visitedNodes: 2 nodes, depth: 1 piles

%%Statistics (ATSE mode): Analysed: 7 states, Reachable: 1 states, Translation: 0.05 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA3

```
role role UE (UE,HSS,MME,ENB: agent, IMSI,UECap,TS,AMF,SQN,DONENAS,DONEAS,HID:text, LKASME,NASCAD,NASIAD,NASALGID, NASALGC:text, NASUPC, LUI,LHI,ASIAD,ASALGID, ASEAD, ASUPEAD,ASALGC :text, HFunc, UIGFunc, HIGFunc, NKGFunc,NKeNBG:hash_func, F1, F2, F3, F4, F5, KDF, AUTHFunc:hash_func, SK:symmetric_key,PKue,PKenb,PKhss,PKmme : public_key, SND_MME,RCV_MME,SND_ENB,RCV_ENB:channel(dy))played_by UE def=local State: nat, RNue, HIMSI,UI,HI,RNmme,RNhss,RAND, AUTH,NKG,MAC, RES,CK, IK, AK,KASME,HNID : text, KNASenc, KNASin,RNenb,KeNB,NKeNB,KRRCCint,KRRCCenc,KRRCCUPen: text const ue_mme_HIMSI,ue_mme_HID,ue_mme_TS,ue_mme_UI,mme_ue_msg_auth_HIMSI,ue_mme_msg_auth_HI, ue_mme_RNue,ue_mme_RNmme,ue_mme_RES,mme_ue_msg_auth_RES,mme_ue_auth_RNmme,ue_mme_msg_auth_NASALGC,ue_mme_auth_RNue, mme_ue_msg_auth_DONENAS, ue_enb_msg_auth_ASALGC,ue_enb_RNenb,ue_enb_DONEAS,ue_enb_msg_auth_DONEAS,ue_enb_auth_RNenb: protocol_id init State := 0 transition 1. State=0 /\ RCV_MME(start) => State:=1 /\ RNue:=new() /\ HIMSI:= HFunc(IMSI.RNue) /\ UI:= UIGFunc(HIMSI.SK.RNue) /\ SND_MME({(HIMSI)_inv (PKue).RNue'.UI'.UECap.TS.PKue.HID}_PKmme) /\ secret(HIMSI'.RNue'.HIMSI, {UE, MME}) /\ secret(HID {UE, MME}) /\ secret(TS ue_mme_TS, {UE, MME}) /\ secret(UI'.ue_mme_UI, {UE, MME}) /\ witness (UE,MME, mme_ue_msg_auth_HIMSI,HIMSI') 4. State=1 /\ RCV_MME({RAND'.AUTH'.RNhss'.RNmme'.HNID'.HI')_inv(PKmme)}_PKue) => State:=2 /\ NKG:= NKGFunc(xor(RNmme', HI').SK) /\ MAC:=F1(NKG'.RAND'.AMF.SQN) /\ RES:=F2(NKG'.RAND') /\ CK:=F3(NKG'.RAND') /\ IK:=F4(NKG'.RAND') /\ AK:=F5(NKG'.RAND') /\ KASME:=KDF(xor(SQN, AK').HNID'.CK'.IK') /\ AUTH:=AUTHFunc(xor(SQN, AK').AMF.MAC') /\ RNue:=new() /\ SND_MME({RNue'.RNmme'.{RES'}_inv(PKue)}_PKmme) /\ request (UE, MME, ue_mme_msg_auth_HI, HI') /\ secret(RNue'.ue_mme_RNue, {UE, MME}) /\ secret(RNmme'.ue_mme_RNmme, {UE, MME}) /\ secret(RES', {UE, MME}) /\ witness (UE,MME, mme_ue_msg_auth_RES,RES') /\ witness (UE,MME, mme_ue_auth_RNmme,RNmme') 6. State=2 /\ RCV_MME({RNue'.{NASALGC}_inv(PKmme)}_PKue) => State:=3 /\ KNASenc:= KDF(LKASME.NASCAD.NASALGID) /\ KNASin:= KDF(LKASME.NASIAD.NASALGID) /\ SND_MME({(DONENAS)_inv(PKue)}_PKmme) /\ request (UE, MME, ue_mme_msg_auth_NASALGC, NASALGC) /\ request (UE, MME, ue_mme_auth_RNue, RNue') /\ secret(DONENAS,ue_mme_DONENAS, {UE, MME}) /\ witness (UE,MME, mme_ue_msg_auth_DONENAS,DONENAS) 9. State=3 /\ RCV_ENB({RNenb'.{ASALGC}_inv(PKenb)}_PKue) => State:=4 /\ KeNB:= KDF(LKASME.NASUPC.LUI.LHI) /\ NKeNB:= NKeNBG(KeNB'.RNenb') /\ KRRCCint:= KDF(NKeNB'.ASIAD,ASALGID) /\ KRRCCenc:= KDF(NKeNB'.ASEAD,ASALGID) /\ KRRCCUPen:= KDF(NKeNB'.ASUPEAD,ASALGID) /\ SND_ENB({RNenb'.(DONEAS)_inv(PKue)}_PKenb) /\ request (UE,ENB,ue_enb_msg_auth_ASALGC, ASALGC) /\ secret(RNenb'.ue_enb_RNenb, {UE, ENB}) /\ secret(DONEAS,ue_enb_DONEAS, {UE, ENB}) /\ witness (UE,ENB, ue_enb_msg_auth_DONEAS,DONEAS) /\ witness (UE,ENB, ue_enb_auth_RNenb,RNenb) end role role UE,HSS,MME,ENB: agent, HVP,MVP:text, LKASME,NASCAD,NASIAD,NASALGID, NASALGC:text, NASUPC, LUI,LHI:text, NIFunc,KDF:hash_func, SKmh,SKem:symmetric_key, PKue,PKenb,PKhss,PKmme : public_key, SND_HSS,RCV_HSS,SND_UE,RCV_UE,SND_ENB,RCV_ENB:channel(dy))played_by MME def=local State: nat, HIMSI,RNue,UI,UECap,TS,HID :text, RNmme,HNID :text, RAND,AUTH,KASME,HI,XRES,RES,RNhss :text, KNASenc, KNASin,DONENAS:text, KeNB:text const mme_ue_msg_auth_HIMSI,mme_hss_HNID,mme_hss_HIMSI, mme_hss_UI,mme_hss_RNmme,mme_hss_RNue,hss_mme_msg_auth_HNID,mme_hss_auth_RNmme,mme_hss_msg_auth_HI,mme_ue_RAND,mme_ue_AUTH,mme_ue_RNhss,mme_ue_RNmme,mme_ue_HNID,mme_ue_HI,ue_mme_msg_auth_HI,mme_ue_msg_auth_RES,mme_ue_auth_RNmme,mme_ue_RNue,mme_ue_NASAL GC,ue_mme_msg_auth_NASALGC,ue_mme_auth_RNue,mme_ue_msg_auth_DONENAS,mme_enb_KeNB,ue_mme_msg_auth_KeNB: protocol_id init State := 0 transition 1. State=0 /\ RCV_UE({(HIMSI')_inv(PKue).RNue'.UI'.UECap.TS.PKue.HID}_PKmme)=> State:=1 /\ HNID:= NIFunc(xor(HVP,MVP)) /\ RNmme:=new() /\ SND_HSS({(RNmme'.RNue'.{HNID'}_inv(PKmme)).HIMSI'.UI')_SKmh) /\ request (MME, UE,mme_ue_msg_auth_HIMSI,HIMSI') /\ secret(HNID', {MME, HSS}) /\ secret(HIMSI', mme_hss_HIMSI, {MME, HSS}) /\ secret(UI'.mme_hss_UI, {MME, HSS}) /\ secret(RNmme', mme_hss_RNmme, {MME, HSS}) /\ secret(RNue'. mme_hss_RNue, {MME, HSS}) /\ witness (MME, HSS, hss_mme_msg_auth_HNID, HNID) 3. State=1 /\ RCV_HSS({(HI')_inv(PKhss).RAND'. AUTH'.KASME'.XRES'.RNhss'.RNmme')_SKmh) => State:=2 /\ HNID:= NIFunc(xor(HVP,MVP)) /\ SND_UE({(RAND'.AUTH'.RNhss'.RNmme'.HNID'.HI')_inv(PKmme)}_PKue) /\ request (MME, HSS, mme_hss_auth_RNmme, RNmme') /\ request (MME, HSS, mme_hss_msg_auth_HI, HI') /\ secret(RAND', mme_ue_RAND, {MME, UE}) /\ secret(AUTH', mme_ue_AUTH, {MME, UE}) /\ secret(RNhss', mme_ue_RNhss, {MME, UE}) /\ secret(RNmme', mme_ue_RNmme, {MME, UE}) /\ secret(HNID', mme_ue_HNID, {MME, UE}) /\ secret(HI', mme_ue_HI, {MME, UE}) /\ witness (MME, UE, ue_mme_msg_auth_HI, HI') 5. State=2 /\ RCV_UE({(RNue'.RNmme'.{RES'}_inv(PKue)}_PKmme)=> State:=3 /\ KNASenc:= KDF(LKASME.NASCAD.NASALGID) /\ KNASin:= KDF(LKASME.NASIAD.NASALGID) /\ SND_UE({RNue'.{NASALGC}_inv(PKmme)}_PKue) /\ request (MME,UE, mme_ue_msg_auth_RES,RES') /\ request (MME,UE, mme_ue_auth_RNmme,RNmme') /\ secret(RNue',mme_ue, RNue, {MME, UE}) /\ secret(NASALGC,mme_ue_NASALGC, {MME, UE}) /\ witness (MME, UE, ue_mme_msg_auth_NASALGC, NASALGC) /\ witness (MME, UE, ue_mme_auth_RNue, RNue') 7. State=3 /\ RCV_UE({(DONENAS)_inv(PKue)}_PKmme)=> State:=4 /\ KeNB:= KDF(LKASME.NASUPC.LUI.LHI) /\ SND_ENB({(KeNB')_inv(PKmme)}_SKem) /\ request (MME,UE, mme_ue_msg_auth_DONENAS,DONENAS) /\ secret(KeNB',mme_enb_KeNB, {MME, ENB}) /\ witness (ENB, ENB, enb_mme_msg_auth_KeNB, KeNB') end role role UE,HSS,MME,ENB: agent, IMSI, HVP, MVP,AMF,SQN:text, HFunc, NIFunc, UIGFunc, HIGFunc, NKGFunc :hash_func, F1, F2, F3, F4, F5, KDF, AUTHFunc :hash_func, SK,SKmh:symmetric_key, PKue,PKenb,PKhss,PKmme : public_key, SND_MME,RCV_MME:channel(dy)) played_by HSS def=local State: nat, RNmme,RNue,HNID, HIMSI,UI:text, RNhss,XHNID,HI,NKG,MAC,RAND,XRES,CK,IK,AK,KASME,AUTH:text const hss_mme_msg_auth_HNID,hss_mme_HI,hss_mme_RAND,hss_mme_AUTH,hss_mme_KASME, hss_mme_XRES, hss_mme_RNhss, hss_mme_RNmme,mme_hss_msg_auth_HI,mme_hss_auth_RNmme: protocol_id init State := 0 transition 2. State=0 /\ RCV_MME({(RNmme'. RNue'.{HNID'}_inv(PKmme)).HIMSI'.UI')_SKmh) => State:=1 /\ RNhss:=new() /\ RAND:=new() /\ HIMSI:= HFunc(IMSI.RNue) /\ XHNID:= NIFunc(xor(HVP,MVP)) /\ UI:= UIGFunc(HIMSI.SK.RNue) /\ HI:= HIGFunc(UI'.SK.RNhss') /\ NKG:= NKGFunc(xor(RNmme', HI').SK) /\ MAC:=F1(NKG'.RAND'.AMF.SQN) /\ XRES:=F2(NKG'.RAND') /\ CK:=F3(NKG'.RAND') /\ IK:=F4(NKG'.RAND') /\ AK:=F5(NKG'.RAND') /\ KASME:=KDF(xor(SQN, AK').XHNID'.CK'.IK') /\ AUTH:=AUTHFunc(xor(SQN, AK').AMF.MAC') /\ SND_MME({(HI')_inv(PKhss). RAND'.AUTH'.KASME'.XRES'.RNhss'.RNmme')_SKmh) /\ request (HSS, MME, hss_mme_msg_auth_HNID, HNID') /\ secret(HI'.hss_mme_HI, {HSS, MME}) /\ secret(RAND'.hss_mme_RAND, {HSS, MME}) /\ secret(AUTH'.hss_mme_AUTH, {HSS, MME}) /\ secret(KASME'.hss_mme_KASME, {HSS, MME}) /\ secret(XRES'.hss_mme_XRES, {HSS, MME}) /\ secret(RNhss'.hss_mme_RNhss, {HSS, MME}) /\ secret(RNmme'.hss_mme_RNmme, {HSS, MME}) /\ witness (HSS, MME, hss_mme_hss_auth_HI, HI') /\ witness (HSS, MME, hss_mme_hss_auth_RNmme, RNmme') end role role UE,HSS,MME,ENB: agent, ASIAD,ASALGID, ASEAD, ASUPEAD,ASALGC:text, NKeNBG,KDF :hash_func, SKem:symmetric_key, PKue,PKenb,PKhss,PKmme : public_key, SND_MME,RCV_MME,SND_UE,RCV_UE:channel(dy))played_by ENB def= local State: nat, NKeNB,KRRCCint,KRRCCenc,KRRCCUPen,RNenb,KeNB,DONEAS:text const enb_mme_msg_auth_KeNB,ue_enb_RNenb,ue_enb_ASALGC,ue_enb_msg_auth_ASALGC,ue_enb_auth_RNenb: protocol_id init State := 0 transition 8. State=0 /\ RCV_MME({(KeNB')_inv(PKmme)}_SKem) => State:=1 /\ RNenb:=new() /\ NKeNB:= NKeNBG(KeNB'.RNenb') /\ KRRCCint:= KDF(NKeNB'.ASIAD,ASALGID) /\ KRRCCenc:= KDF(NKeNB'.ASEAD,ASALGID) /\ KRRCCUPen:= KDF(NKeNB'.ASUPEAD,ASALGID) /\ SND_UE({(RNenb'.{ASALGC}_inv(PKenb)}_PKue) /\ request (ENB,MME, enb_mme_msg_auth_KeNB, KeNB') /\ secret(RNenb'.enb_ue_RNenb, {ENB, UE}) /\ secret(ASALGC',enb_ue_ASALGC, {ENB, UE}) /\ witness (ENB, UE, ue_enb_msg_auth_ASALGC, ASALGC) 10. State=1 /\ RCV_UE({(RNenb'.{DONEAS}_inv(PKue)}_PKenb) => State:=2 /\ request (ENB,UE, enb_ue_msg_auth_DONEAS,DONEAS) /\ request (ENB,UE,ue_enb_auth_RNenb,RNenb') end role role session1(UE,MME,HSS,ENB:agent, IMSI,UECap,TS,AMF,SQN,DONENAS,DONEAS,HID:text, LKASME,NASCAD,NASIAD,NASALGID, NASALGC:text, NASUPC, LUI,LHI,ASIAD,ASALGID, ASEAD, ASUPEAD,ASALGC :text, RNue, HIMSI,UI,HI,RNmme,RNhss,RAND, AUTH,NKG,MAC,RES,CK, IK, AK,KASME,HNID : text, KNASenc, KNASin,KeNB,NKeNB,KRRCCint,KRRCCenc,KRRCCUPen: text, HVP,MVP,XHNID,XRES,RNenb:text, HFunc, UIGFunc, HIGFunc, NKGFunc,NKeNBG:hash_func, NIFunc, F1, F2, F3, F4, F5, KDF, AUTHFunc:hash_func, PKue,PKenb,PKhss,PKmme : public_key, SK,SKmh,SKem:symmetric_key ) def=local SND_MME1,RCV_MME1,SND_ENB1,RCV_ENB1, SND_HSS2,RCV_HSS2,SND_UE2,RCV_UE2,SND_ENB2,RCV_ENB2, SND_MME3,RCV_MME3, SND_MME4,RCV_MME4,SND_UE4,RCV_UE4 : channel(dy) composition role UE(UE, MME, HSS,ENB,IMSI,UECap,TS,AMF,SQN,DONENAS,DONEAS,HID, LKASME,NASCAD,NASIAD,NASALGID, NASALGC,NASUPC,LUI,LHI,ASIAD,ASALGID,ASEAD,ASUPEAD,ASALGC,HFunc,UIGFunc,HIGFunc,NKGFunc,NKeNBG,F1,F2,F3,F4,F5,KDF,AUTHFunc,S K,PKue,PKenb,PKhss,PKmme,SND_MME1, RCV_MME1,SND_ENB1,RCV_ENB1)/role_MME(UE,MME,HSS,ENB,HVP,MVP,LKASME,NASCAD,NASIAD,NASALGID,NASALGC,NASUPC,LUI,LHI,NIFu nc,KDF,SKmh,SKem,PKue,PKenb,PKhss, PKmme,SND_HSS2,RCV_HSS2,SND_UE2,RCV_UE2,SND_ENB2,RCV_ENB2)/role_HSS(UE,MME,HSS,ENB,IMSI,HVP,MVP,AMF,SQN,HFunc,NIFunc,UIGFu nc,HIGFunc,NKGFunc, F1,F2,F3,F4,F5,KDF, AUTHFunc,SK,SKmh,PKue,PKenb,PKhss,PKmme,SND_MME3,RCV_MME3)/role_ENB(UE,MME,HSS,ENB,ASIAD,ASALGID,ASEAD,ASUPEAD,ASALGC,NK eNBG,KDF,SKem,PKue,PKenb, PKhss, PKmme, SND_MME4,RCV_MME4,SND_UE4,RCV_UE4) end role role environment() def= const agent,imsi,uecap,ts,amf,sqn,donenas,doneas,hid:text, lkasme,nascad,nasiad,nasalgid, nasalgc:text,nasupc, lui,hi,asiad,asalgid, asead, asupead,asalgc :text, rnue, himsi,ui,hi,mmme,mhss,rand, auth,nkg,mac,sqn,res,ck, ik, ak,kasme,hnid : text, knasenc, knasin,kenb,nkenb,krrccint,krrccenc,krrccupen: text, hvp,mvp,xhnid,xres,rnenb : text, hfunc, uigfunc, higfunc, nkgfunc,nkenbg,nifunc, f1, f2, f3, f4, f5, kdf, authfunc:hash_func, pkue,pkenb,pkhss,pkmme:public_key, sk,skmh,skem:symmetric_key, ue_mme_HIMSI,ue_mme_HID,ue_mme_TS,ue_mme_UI,mme_ue_msg_auth_HI, ue_mme_RNue,ue_mme_RNmme,ue_mme_RES,mme_ue_msg_auth_RES,mme_ue_auth_RNmme,ue_mme_msg_auth_NASALGC,ue_mme_auth_RNue,ue_mme_DO NENAS,ue_msg_auth_DONENAS,ue_enb_msg_auth_ASALGC,ue_enb_RNenb,ue_enb_DONEAS,ue_enb_msg_auth_DONEAS,ue_enb_auth_RNenb,mme_hss_H
```

HNID,mme_hss_HIMSI,mme_hss_UI,
mme_hss_RNnme,mme_hss_RNue,hss_mme_msg_auth_HNID,mme_hss_auth_RNnme,mme_hss_msg_auth_HI,mme_ue_RAND,mme_ue_AUTH,mme_ue_RNhss,
mme_ue_RNmme, mme_ue_HNID,mme_ue_HI,ue_mme_msg_auth_HI,mme_ue_RNue,mme_ue_NASALGC,mme_enb_KeNB,enb_mme_msg_auth_KeNB,
hss_mme_HI,hss_mme_RAND,hss_mme_AUTH, hss_mme_KASME,hss_mme_XRES,hss_mme_RNhss,hss_mme_RNmme,enb_ue_RNenb,enb_ue_ASALGC:
protocol_id intruder_knowledge = {hfunc, uigfunc, higfunc, nkgfunc,nkenbg,nifunc, f1, f2, f3, f4, f5, kdf, authfunc, pkue,pkenb,pkhss,pkmmme} composition
session1(ue,mme,hss,enb,imsi,uecap,ts,amf,sqn,donenas,doneas,hid,lkasme,nascd,nasiad,nasalgid,nasalgc,
nasupc,lui,lhi,asiad,asalgid,asead,asupcad,asalgc,rnue,himsi,ui,hi,rmnme,rnhss,rand,auth,nkg,mac,sqn,res,ck,ik,ak,kasme,hnid,knasenc,knasin,kenb,nkenb,krrcint,krrcenc,
krrcupen, hvp,mvp, xhnid,xres,hfunc, uigfunc, higfunc, nkgfunc,nkenbg,nifunc,f1, f2, f3, f4,f5,kdf,authfunc, pkue,pkenb,pkhss,pkmmme,sk,skmh,skem) end role goal
authentication_on mme_ue_msg_auth_HIMSI authentication_on ue_mme_msg_auth_HI authentication_on mme_ue_msg_auth_RES authentication_on
mme_ue_auth_RNmme authentication_on ue_mme_msg_auth_NASALGC authentication_on ue_mme_auth_RNue authentication_on mme_ue_msg_auth_DONENAS
authentication_on ue_enb_msg_auth_ASALGC authentication_on enb_ue_msg_auth_DONEAS authentication_on enb_ue_auth_RNenb authentication_on
hss_mme_msg_auth_HNID authentication_on mme_hss_auth_RNnme authentication_on mme_hss_msg_auth_HI authentication_on ue_mme_msg_auth_HI
authentication_on enb_mme_msg_auth_KeNB secrecy_of ue_mme_HIMSI secrecy_of ue_mme_HID secrecy_of ue_mme_TS secrecy_of ue_mme_UI secrecy_of
ue_mme_RNue secrecy_of ue_mme_RNmme secrecy_of ue_mme_RES secrecy_of ue_mme_DONENAS secrecy_of ue_enb_RNenb secrecy_of ue_enb_DONEAS
secrecy_of mme_hss_HNID secrecy_of mme_hss_HIMSI secrecy_of mme_hss_UI secrecy_of mme_hss_RNmme secrecy_of mme_hss_RNue secrecy_of
mme_ue_RAND secrecy_of mme_ue_AUTH secrecy_of mme_ue_RNhss secrecy_of mme_ue_RNmme secrecy_of mme_ue_HNID secrecy_of mme_ue_HI secrecy_of
mme_ue_RNue secrecy_of mme_ue_NASALGC secrecy_of mme_enb_KeNB secrecy_of hss_mme_HI secrecy_of hss_mme_RAND secrecy_of hss_mme_AUTH
secrecy_of hss_mme_KASME secrecy_of hss_mme_XRES secrecy_of hss_mme_RNhss secrecy_of hss_mme_RNmme secrecy_of enb_ue_RNenb secrecy_of
enb_ue_ASALGC end goal environment()

%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.12s, visitedNodes: 2 nodes, depth: 1 piles

%Statistics (ATSE mode): Analysed: 7 states, Reachable: 1 states, Translation: 0.04 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA4.1.1

role role_UE (UE,HSS,MME: agent, IMSI,TS, AMF,SQN :text, UIGFunc,NIGFunc,NKGFunc, F1, F2, F3, F4, F5, KDF, AUTHFunc:hash_func, SK:symmetric_key,
PKue,PKmme, PKhss : public_key, SND_MME,RCV_MME:channel(dy))played_by UE def-local State:nat, UI, RNue,RNmme,RNhss,NI,UKG, MAC,RES,CK, IK,
AK,KASME,AUTN, RAND,HNID : text const ue_mme_IMSI, ue_mme_RNue, ue_mme_TS,ue_mme_UI,mme_ue_msg_auth_IMSI, ue_mme_RNmme, ue_mme_RES,
ue_mme_UGK_msg_auth_NI, mme_ue_msg_auth_RES, mme_ue_auth_RNmme: protocol_id init State := 0 transition 1. State=0/RCV_MME(start) => State':=1/
RNue':=new()/ UI':= UIGFunc(IMSI.xor(RNue', SK)) ^ SND_MME({(IMSI)_inv(PKue).RNue'.UI'.PKue.TS})_PKmme)/ secret(IMSI, ue_mme_IMSI, {UE, MME})/
secret(RNue', ue_mme_RNue, {UE, MME}) ^ secret(TS, ue_mme_TS, {UE, MME}) ^ secret(UI', ue_mme_UI, {UE, MME}) ^ witness (UE,MME) ^
secret(RNue', ue_mme_RNue, {UE, MME}) ^ secret(TS, ue_mme_TS, {UE, MME}) ^ secret(UI', ue_mme_UI, {UE, MME}) ^ witness (UE,MME) ^
mme_ue_msg_auth_IMSI,IMSI) 4. State=1 ^ RCV_MME((RAND'.AUTN'.RNhss'.RNmme'.HNID'.NI')_inv(PKmme))_PKue => State':=2 ^ NI':=
NIGFunc(RNhss'.SK.UI) ^ UKG':= NKGFunc(SK.xor(RNmme', NI')) ^ MAC':=F1(UKG'.RAND'.AMF.SQN) ^ RES':=F2(UKG'.RAND') ^ CK':=F3(UKG'.RAND') ^
IK':=F4(UKG'.RAND') ^ AK':=F5(UKG'.RAND') ^ KASME':=KDF(xor(SQN, AK)).HNID'.CK'.IK'.UKG) ^ AUTN':=AUTHFunc(xor(SQN, AK)).AMF.MAC) ^
SND_MME({(RNmme'.{RES'}_inv(PKue))_PKmme) ^ secret(RNmme',ue_mme_RNmme, {UE, MME}) ^ secret(RES',ue_mme_RES, {UE, MME}) ^ request (UE,
MME, ue_mme_UGK_msg_auth_NI, NI') ^ witness (UE,MME, mme_ue_msg_auth_RES,RES') ^ witness (UE,MME, mme_ue_auth_RNmme,RNmme') end role role
role_MME(UE,HSS,MME: agent, HVP,MVP:text,NIFunc:hash_func, SKmh:symmetric_key, PKue,PKmme, PKhss : public_key,
SND_HSS,RCV_HSS,SND_UE,RCV_UE:channel(dy)) played_by MME def-local State:nat, RNue,IMSI,UI,TS, HNID,
RAND,AUTN,RNhss,RNhss,NI,XRES,RES,KASME:text
const mme_hss_HNID,mme_hss_IMSI,mme_hss_UI,mme_hss_RNmme,mme_hss_RNue,mme_ue_msg_auth_IMSI,hss_mme_msg_auth_HNID,mme_ue_RAND,mme_ue_A
UTN,mme_ue_RNhss, mme_ue_RNmme, mme_ue_HNID,mme_ue_NI,mme_hss_auth_RNmme,mme_hss_msg_auth_NI, ue_mme_msg_auth_NI,
mme_ue_msg_auth_RES,mme_ue_auth_RNmme:protocol_id init State := 0 transition 1. State=0/RCV_UE({(IMSI)_inv(PKue).RNue'.UI'.PKue.TS})_PKmme
=>|State':=1 /HNID':= NIFunc(xor(HVP,MVP)) ^ RNmme':=new() ^ SND_HSS({(RNmme'.RNue'.{HNID'}_inv(PKmme).IMSI.UI')_SKmh) ^
secret(HNID',mme_hss_HNID, {MME, HSS}) ^ secret(IMSI, mme_hss_IMSI, {MME, HSS}) ^ secret(UI', mme_hss_UI, {MME, HSS}) ^ secret(RNmme',
mme_hss_RNmme, {MME, HSS}) ^ secret(RNue', mme_hss_RNue, {MME, HSS}) ^ request (MME,UE, mme_ue_msg_auth_IMSI,IMSI) ^ witness (MME, HSS,
hss_mme_msg_auth_HNID,HNID')3.State=1/RCV_HSS({(NI')_inv(PKhss).RAND'.AUTN'.KASME'.XRES'.RNhss'.RNmme')_SKmh}>=>State':=2/SND_UE({(RAN
D'.AUTN'.RNhss'. RNmme'.HNID'. {NI'}_inv(PKmme))_PKue) ^ secret(RAND', mme_ue_RAND, {MME, UE}) ^ secret(AUTN', mme_ue_AUTN, {MME, UE}) ^
secret(RNhss', mme_ue_RNhss, {MME, UE}) ^ secret(RNmme', mme_ue_RNmme, {MME, UE}) ^ secret(HNID', mme_ue_HNID, {MME, UE}) ^ secret(NI', mme_ue_NI,
{MME, UE}) ^ request (MME, HSS, mme_hss_auth_RNmme, RNmme') ^ request (MME, HSS, mme_hss_msg_auth_NI, NI') ^ witness (MME, UE, ue_mme
msg_auth_NI, NI') 5.State=2/RCV_UE({(RNmme'.{RES'}_inv(PKue))_PKmme) => State':=3 ^ request (MME,UE, mme_ue_msg_auth_RES,RES') ^ request
(MME,UE, mme_ue_auth_RNmme,RNmme') end role role role_HSS(UE,HSS,MME: agent, IMSI, HVP, MVP,AMF,SQN:text, UIGFunc,NIGFunc, NIFunc,NKGFunc,
F1, F2, State:nat, RNmme,RNue,HNID, UI,RNhss,NI,UKG,MAC,RAND,XRES,CK,IK,AK,KASME,AUTN:text const hss_mme_NI, hss_mme_RAND,
hss_mme_AUTN, hss_mme_KASME, hss_mme_XRES, hss_mme_RNhss, hss_mme_RNmme,
hss_mme_msg_auth_HNID,mme_hss_auth_RNmme,mme_hss_msg_auth_NI:protocol_id init State := 0 transition 2. State=0 ^
RCV_MME({(RNmme'.RNue'.{HNID'}_inv(PKmme).IMSI.UI')_SKmh) =>|State':=1 ^ RNhss':=new() ^ RAND':=new() ^ HNID':= NIFunc(xor(HVP,MVP)) ^ UI':=
UIGFunc(IMSI.xor(RNue', SK)) ^ NI':= NIGFunc(RNhss'.SK.UI) ^ UKG':= NKGFunc(SK.xor(RNmme', NI')) ^ MAC':=F1(UKG'.RAND'.AMF.SQN) ^
XRES':=F2(UKG'.RAND') ^ CK':=F3(UKG'.RAND') ^ IK':=F4(UKG'.RAND') ^ AK':=F5(UKG'.RAND') ^ KASME':=KDF(xor(SQN, AK)).HNID'.CK'.IK'.UKG) ^
AUTN':=AUTHFunc(xor(SQN, AK)).AMF.MAC) ^ SND_MME({(NI')_inv(PKhss).RAND'.AUTN'. KASME'.XRES'.RNhss'.RNmme')_SKmh) ^
secret(NI',hss_mme_NI, {HSS, MME}) ^ secret(RAND',hss_mme_RAND, {HSS, MME}) ^ secret(AUTN',hss_mme_AUTN, {HSS, MME}) ^
secret(KASME',hss_mme_KASME, {HSS, MME}) ^ secret(XRES',hss_mme_XRES, {HSS, MME}) ^ secret(RNhss',hss_mme_RNhss, {HSS, MME}) ^
secret(RNmme',hss_mme_RNmme, {HSS, MME}) ^ request (HSS, MME, mme_hss_msg_auth_HNID, HNID) ^ witness (HSS, MME, mme_hss_auth_RNmme,
RNmme') ^ witness (HSS, MME, mme_hss_msg_auth_NI, NI')end role role session1(UE, MME,HSS:agent, IMSI,TS, AMF,SQN,UI, RNue,RNmme,RNhss,NI,UKG,
MAC,RES,CK, IK, AK,KASME,AUTN, RAND,HNID:text, HVP,MVP, XRES:text, UIGFunc,NIGFunc,NKGFunc, F1, F2, F3, F4, F5, KDF,
AUTHFunc,NIFunc:hash_func, PKue,PKhss,PKmme : public_key, SK,SKmh:symmetric_key) def-local SND_MME1,RCV_MME1,
SND_HSS2,RCV_HSS2,SND_UE2,RCV_UE2, SND_MME3,RCV_MME3 : channel(dy) composition role_UE(UE, MME, HSS,IMSI,TS,
AMF,SQN,UIGFunc,NIGFunc,NKGFunc, F1, F2, F3, F4, F5, KDF, AUTHFunc, SK,PKue,PKmme, PKhss, SND_MME1,RCV_MME1) /role_MME(UE, MME,
HSS,HVP,MVP,NIFunc,SKmh,PKue,PKmme,PKhss,SND_HSS2,RCV_HSS2,SND_UE2,RCV_UE2) /role_HSS(UE, MME, HSS,IMSI, HVP,
MVP,AMF,SQN,UIGFunc,NIGFunc, NIFunc, NKGFunc, F1, F2, F3, F4, F5, KDF, AUTHFunc,SK,SKmh, PKue,PKmme, PKhss,SND_MME3,RCV_MME3) end role
role environment() const ue:agent,mme:agent,hss:agent, imsi,ts, amf,sqn,ui, rnue,rmnme,rnhss,ni,ukg, mac,res,ck, ik, ak,kasme,autn, rand,hnid,hvp,mvp,
xres:text, uigfunc,nigfunc,nkgfunc, f1, f2, f3, f4, f5, kdf, authfunc,nifunc:hash_func, pkue,pkhss,pkmmme:public_key, sk,skmh:symmetric_key, ue_mme_IMSI,
ue_mme_RNue, ue_mme_TS,ue_mme_UI,mme_ue_msg_auth_IMSI,ue_mme_RNmme, ue_mme_RES,ue_mme_msg_auth_NI, mme_ue_auth_RES,
mme_ue_auth_RNmme, mme_hss_HNID,mme_hss_IMSI,mme_hss_UI,mme_hss_RNmme, mme_hss_RNue, hss_mme_msg_auth_HNID,mme_ue_RAND,
mme_ue_AUTN, mme_ue_RNhss,mme_ue_RNmme, mme_ue_HNID,mme_ue_NI,mme_hss_auth_RNmme,mme_hss_msg_auth_NI,hss_mme_RAND,
hss_mme_AUTN,hss_mme_KASME, hss_mme_XRES, hss_mme_RNhss,hss_mme_RNmme :protocol_id intruder_knowledge = {uigfunc,nigfunc,nkgfunc, f1, f2, f3, f4,
f5, kdf, authfunc,nifunc,pkue,pkhss,pkmmme} composition
session1(ue,mme,hss,imsi,ts,amf,sqn,ui, rnue,rmnme,rnhss,ni,ukg,mac,res,ck, ik, ak,kasme,autn,rand,hnid,hvp,mvp, xres,uigfunc, nigfunc,nkgfunc, f1, f2, f3, f4, f5, kdf,
authfunc,nifunc, pkue,pkhss,pkmmme,sk,skmh) end role goal authentication_on mme_ue_msg_auth_IMSI authentication_on ue_mme_msg_auth_NI authentication_on
mme_ue_msg_auth_RES authentication_on mme_ue_auth_RNmme authentication_on hss_mme_msg_auth_HNID authentication_on mme_hss_auth_RNmme
authentication_on mme_hss_msg_auth_NI secrecy_of ue_mme_IMSI secrecy_of ue_mme_RNue secrecy_of ue_mme_TS secrecy_of ue_mme_UI secrecy_of
ue_mme_RNmme secrecy_of ue_mme_RES secrecy_of mme_hss_HNID secrecy_of mme_hss_IMSI secrecy_of mme_hss_UI secrecy_of mme_hss_RNmme secrecy_of
mme_hss_RNue secrecy_of mme_ue_RAND secrecy_of mme_ue_AUTN secrecy_of mme_ue_RNhss secrecy_of mme_ue_RNmme secrecy_of mme_ue_HNID
secrecy_of mme_ue_NI secrecy_of hss_mme_NI secrecy_of hss_mme_RAND secrecy_of hss_mme_AUTN secrecy_of hss_mme_KASME secrecy_of hss_mme_XRES
secrecy_of hss_mme_RNhss secrecy_of hss_mme_RNmme end goal environment()

%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.03s, visitedNodes: 2 nodes, depth: 1 piles

%Statistics (ATSE mode): Analysed: 5 states, Reachable: 1 states, Translation: 0.01 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA4.1.2

role role_UE (UE,HSS,MME: agent, UI, IMSI, KASME,NI,RES,AUTN, RAND:text, NUIGFunc,NNIGFunc, KDF, UPKFunc:hash_func, SK:symmetric_key,
PKue,PKmme, PKhss : public_key, SND_MME,RCV_MME:channel(dy))played_by UE def-local State:nat,
RNue,NUI,NNI,RNmme,RNhss,UKASME,UPK,NRES,UAUTN, URAND: text const ue_mme_IMSI, ue_mme_RNue, ue_mme_NUI,
mme_ue_msg_auth_NUI,ue_mme_RNmme, ue_mme_NRES, ue_mme_msg_auth_UAUTN, ue_mme_auth_RNue,mme_ue_msg_auth_NRES, mme_ue_auth_RNmme:
protocol_id init State := 0 transition 1. State=0 ^ RCV_MME(start) => State':=1 ^ RNue':=new() ^ NUI':= NUIGFunc(UI.RNue') ^
SND_MME({(NUI')_inv(PKue).RNue'.IMSI)_PKmme) ^ secret(IMSI, ue_mme_IMSI, {UE, MME}) ^ secret(RNue',ue_mme_RNue, {UE, MME}) ^ secret(NUI',
ue_mme_NUI, {UE, MME}) ^ witness (UE,MME, mme_ue_msg_auth_NUI) 4. State=1 ^
RCV_MME({(RNmme'.RNue'.{UAUTN'}_inv(PKmme).URAND')_PKue) =>|State':=2 ^ NNT':= NNIGFunc(NI.NUI.SK) ^ UPK':= UPKFunc(NUI.RNmme') ^
UKASME':= KDF(KASME.UPK) ^ NRES':= KDF(RES.UPK) ^ UAUTN':= KDF(AUTN.UPK) ^ URAND':= KDF(RAND.UPK) ^
SND_MME({(RNmme'.{NRES'}_inv(PKue))_PKmme) ^ secret(RNmme',ue_mme_RNmme, {UE, MME}) ^ secret(NRES',ue_mme_NRES, {UE, MME}) ^ request
(UE,MME, ue_mme_msg_auth_UAUTN, UAUTN') ^ request (UE,MME, ue_mme_auth_RNue, RNue') ^ witness (UE,MME, mme_ue_msg_auth_NRES,NRES') ^
witness (UE,MME, mme_ue_auth_RNmme,RNmme') end role role role_MME(UE,HSS,MME: agent, UI,AUTN, RAND,KASME,XRES:text,
NUIGFunc,UPKFunc,KDF:hash_func, SKmh:symmetric_key, PKue,PKmme, PKhss : public_key, SND_HSS,RCV_HSS,SND_UE,RCV_UE:channel(dy))played_by
MME def-local State:nat, RNue,RNmme,RNhss,NUI,UPK,IMSI,UAUTN, URAND,UKASME,NXRES,NRES:text const mme_hss_HSS, mme_hss_RNue,
mme_hss_NUI, mme_hss_IMSI, mme_ue_RNmme, mme_ue_RNue, mme_ue_UAUTN, mme_ue_URAND, mme_ue_msg_auth_NUI,ue_mme_msg_auth_UAUTN,
ue_mme_auth_RNue, hss_mme_msg_auth_NUI, me_ue_msg_auth_NRES, mme_ue_auth_RNmme:protocol_id init State := 0 transition 1. State=0 ^
RCV_UE({(NUI')_inv(PKue).RNue'.IMSI)_PKmme) =>|State':=1 ^ RNmme':=new() ^ NUI':= NUIGFunc(UI.RNue') ^ UPK':= UPKFunc(NUI'.RNmme') ^ UAUTN':=

%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.01s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 3 states, Reachable: 1 states, Translation: 0.00 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA4.2.1

```

role UE, AP, WAAA, LAAA, HSS: agent, IM$IHND, TS, AT, NINNI, RES, text, KDF:hash_func, PKue, PKAp : public_key,
SND_AP, RCV_AP: channel(dy) played by UE def=local State:nat, RNue, UI, RNlis, RNap, NRES, MK, AKAs: text const tucn ue_ap_RNue, ue_ap_HNID, ue_ap_AT,
ue_ap_NINNI, ue_ap_IMSI, ue_ap_RNAP, ue_ap_NRES, ap_ue_msg_auth_NRES, ap_ue_msg_auth_NINNI: protocol_id init State := 0 transition 1. State=0 /
RCV_AP(start)=> State:=1 / RNue:=new() / SND_AP((RNue).HNID,AT,{NINNI}.inv(PKue),IMSI)_PKAp / secret(RNue,ue_ap_RNue,{UE,AP})_secret(HNID,{UE,AP},
ue_ap_HNID,{UE,AP})_secret(AT,ue_ap_AT,{UE,AP})_secret(NINNI,ue_ap_NINNI,{UE,AP})_secret(IMSI,ue_ap_IMSI,{UE,AP})_witness
(UE,AP,ue_ap_msg_auth_NINNI,NINNI) 9. State=1 / RCV_AP((RNLis).UI.RNAP)_PKue => State:=2 / NRES:= KDF(RES.RNlis) /
SND_AP((RNAP)[NRES].inv(PKue))_PKAp / secret(RNAP,ue_ap_RNAP,{UE,AP})_secret(NRES,ue_ap_NRES,{UE,AP})_witness (UE,AP,
ap_ue_msg_auth_NRES,NRES) 13. State=2 / RCV_AP((AKAs)_PKue) => State:=3 / MK:= KDF(UI.NRES.RNlis) end role role
ap_ue, AP, WAAA, LAAA, HSS: agent, APID: text, SKapws: symmetric_key, PKue, PKAp : public_key,
SND_UE, RCV_UE, SND_WAAA, RCV_WAAA: channel(dy) played by AP def=local State:nat,
RNue, HNID, AT, TS, NINNI, IMSI, RNlis, UI, RNap, NRES, MK, AKAs: text const ap_waaa_AT, ap_waaa_HNID, ap_waaa_NINNI, ap_waaa_IMSI,
ap_waaa_APID,
ap_ue_msg_auth_NINNI, waaa_ap_msg_auth_NINNI, ap_ue_RNlis, ap_ue_UI, ap_ue_RNAP, ap_waaa_NRES, ap_ue_msg_auth_NRES,
aaaa_ap_msg_auth_NRES, ap_ue_AKAs: protocol_id init State := 0 transition 2. State=0 / RCV_UE((RNue).HNID,AT,{NINNI}.inv(PKue),IMSI)_PKAp => State:=1 /
SND_WAAA({AT,HNID},{NINNI}.inv(PKue),IMSI,APID)_SKapws / secret(AT,ap_waaa_AT,{AP,WAAA})_secret(HNID,ap_waaa_HNID,{AP,WAAA})_
secret(NINNI,ap_waaa_NINNI,{AP,WAAA})_secret(IMSI,ap_waaa_IMSI,{AP,WAAA})_secret(APID,ap_waaa_APID,{AP,WAAA})_request (AP,UI,
ap_ue_msg_auth_NINNI,NINNI) / witness (AP,WAAA,aaaa_ap_msg_auth_NINNI,NINNI) 8. State=1 / RCV_WAAA({RNlis.PKue,UI}_SKapws) => State:=2 /
RNAP:=new() / SND_UE((RNLis.UI.RNAP)_PKue) / secret(RNlis'_ap_ue_RNlis,{AP,UE})_secret(UI,ap_ue_UI,{AP,UE})_secret(RNAP,ap_ue_RNAP,{AP,UE})
10. State=2 / RCV_UE((RNAP)[NRES].inv(PKue))_PKAp => State:=3 / SND_WAAA({NRES}.inv(PKue))_SKapws / secret(NRES,ap_waaa_NRES,{AP,
WAAA})_request (AP,UE,ap_ue_msg_auth_NRES,NRES)' / witness (AP,WAAA,aaaa_ap_msg_auth_NRES,NRES) 12. State=3 /
RCV_WAAA((AKAS,MK)_SKapws) => State:=4 / SND_UE({AKAS}_PKue) / secret(AKAs,ap_ue_AKAs,{AP,UE}) end role role WAAAA, UE, AP, WAAA,
LAAA, HSS: agent, AKAs: text, KDF: hash_func, SKapws, SKwls: symmetric_key PKue, PKAp: public_key, SND_AP, RCV_AP, SND_LAAA, RCV_LAAA: channel(dy)
played by WAAA def=local State:nat, AT, HNID, NINNI, IMSI, APID, NXRES, RNlis, UI, NRES, MK: text const waaa_laaa_AT, waaa_laaa_HNID, waaa_laaa_NINNI,
waaa_laaa_IMSI, waaa_ap_msg_auth_NINNI, laaa_waaa_msg_auth_NINNI, waaa_ap RAND, waaa_ap PKue, waaa_ap AKAs, waaa_ap MK,
waaa_ap msg_auth_NRES: protocol_id init State := 0 transition 3. State=0 / RCV_AP({AT,HNID},{NINNI}.inv(PKue),IMSI,APID)_SKapws => State:=1 /
SND_LAAA({AT,HNID},{NINNI}.inv(PKue),IMSI)_SKwls / secret(AT,waaa_laaa_AT,{WAAAA,LAAA})_secret(HNID,waaa_laaa_HNID,{WAAAA,LAAA})_
secret(NINNI,waaa_laaa_NINNI,{WAAAA,LAAA})_secret(IMSI,waaa_laaa_IMSI,{WAAAA,LAAA})_request (WAAAA,AP, waaa_ap msg_auth_NINNI, NINNI) /
witness (WAAAA,AP, laaa_waaa_msg_auth_NINNI, NINNI) 7. State=1 / RCV_LAAA({NXRES.RNlis'.PKue,UI}_SKwls) => State:=2 /
SND_AP((RNlis'.PKue,UI)_SKapws) / secret(RNlis'_waaa_ap RAND,{WAAAA,AP})_secret(PKue,waaa_ap PKue,{WAAAA,AP})_secret(UI,waaa_ap UI,
{WAAAA,AP}) 11. State=2 / RCV_AP({NRES}.inv(PKue))_SKapws => State:=3 / MK:= KDF(UI.NRES.RNlis) / SND_AP((AKAs,MK')_SKapws) /
secret(AKAs, waaa_ap AKAs, {WAAAA,AP})_secret(MK', waaa_ap MK, {WAAAA,AP})_request (WAAAA,AP, waaa_ap msg_auth_NRES,NRES') end role role
LAAAUAE, UE, AP, WAAA, LAAA, HSS: agent, KDF: hash_func, SKwls: symmetric_key,
PKue, PKhss, PKlis: public_key, SND_WAAA, RCV_WAAA, SND_HSS, RCV_HSS: channel(dy) played by LAAA def=local State:nat,
AT, HNID, NINNI, IMSI, RNlis, XRES, UI, NXRES: text const laaa_HSS, RNlis, laaa_waaa_msg_auth_NINNI, hss_laaa_msg_auth_IMSI, laaa_waaa_XRES,
laaa_waaa_RNlis, laaa_waaa_PKue, laaa_waaa_UI, laaa_hss_msg_auth_NINNI: protocol_id init State := 0 transition 4. State=0 /
RCV_WAAA({AT,HNID,{NINNI}.inv(PKue),IMSI)_SKwls) => State:=1 / LAAA:=new() / RNlis': new() / SND_HSS({RNlis'.IMSI}_inv(PKlis))_PKhss
secret(RNlis',laaa_hss_RNlis,{LAAA,HSS})_secret(IMSI,laaa_hss_IMSI,{LAAA,HSS})_request (LAAA,WAAA, laaa_waaa_msg_auth_NINNI, NINNI) / witness
(LAAA,HSS,hss_laaa_msg_auth_IMSI,IMSI) 6. State=1 / RCV_HSS({RNlis'.XRES.PKue,{NINNI}.inv(PKhss),UI}_PKlis) => State:=2 /
NXRES': KDF(XRES.RNlis') / SND_WAAA({NXRES'.RNlis'.PKue,UI}_SKwls) / secret(NXRES', laaa_waaa_NXRES, {LAAA,WAAA})_secret(RNlis'
laaa_waaa_RNlis, {LAAA,WAAA})_secret(PKue, laaa_waaa_PKue, {LAAA,WAAA}) / secret(UI, laaa_waaa_UI, {LAAA,WAAA})_request (LAAA,HSS,
laaa_hss_msg_auth_NINNI, NINNI) end role role HSS(UE,AP,WAAA,LAAA,HSS: agent, XRES,UI,NINNI:text, PKue,PKhss,PKlis : public_key,
SND_LAAA,RCV_LAAA:channel(dy))played by HSS def= local State:nat, RNlis,IMSI:text const hss_laaa_RNlis, hss_laaa_XRES, hss_laaa_PKue, hss_laaa_NINNI,
hss_laaa_UI, hss_laaa_msg_auth_IMSI, laaa_hss_msg_auth_NINNI: protocol_id init State := 0 transition 5. State=0 / RCV_LAAA({RNlis'.IMSI}_inv(PKlis))_PKhss
=> State:=1 / SND_LAAA({RNlis'.XRES.PKue,{NINNI}.inv(PKhss),UI}_PKlis) / secret(RNlis',hss_laaa_RNlis,{HSS,LAAA}) / secret(XRES,hss_laaa_XRES,{HSS,
LAAA})_secret(PKue,hss_laaa_PKue,{HSS,LAAA})_secret(NINNI,hss_laaa_NINNI,{HSS,LAAA}) / secret(UI,hss_laaa_UI,{HSS,LAAA})_request (HSS,LAAA,
hss_laaa_msg_auth_NINNI,IMSI) / witness (HSS,LAAA, laaa_hss_msg_auth_NINNI, NINNI) end role role session1(UE,AP,WAAA,LAAA,HSS:agent,
IMSI,HNID,TS,AT,NINNI,RES,RNue,UI,RNlis,RNAP,NRES,MK,AKAs, APID, NXRES,XRES:text, KDF:hash_func, PKue,PKAp, PKhss,PKlis :
public_key,SKapws,SKwls:symmetric_key)def=local
SND_AP1,RCV_AP1,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3,SND_WAAA4, RCV_WAAA4,
SND_HSS4,RCV_HSS4,SND_LAAA5,RCV_LAAA5:channel(dy) composition role UE(UE,AP,WAAA,LAAA,HSS,IMSI,HNID,TS,AT, NINNI,RES,KDF,
PKue,PKAp,SND_AP1,RCV_AP1)
/role AP(UE,AP,WAAA,LAAA,HSS,APID,SKapws,PKue,PKAp,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2)/role WAAA(UE,AP,WAAA,LAAA,HSS,AK
As,KDF,SKapws,SKwls,PKue,PKAp,
SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3)/role_LAAA(UE,AP,WAAA,LAAA,HSS,KDF,SKwls,PKue,PKhss,PKlis,SND_WAAA4,RCV_WAAA4,SND_HS
S4,RCV_HSS4)/role_HSS(UE,AP,WAAA,LAAA,HSS,XRES,UI,NINNI,PKue,PKhss,PKlis,SND_LAAA5,RCV_LAAA5) end role role environment() def=const
ue:agent,ap:agent,laaa:agent,hss:agent,imsi,hnid,ts,at, ninni,res,rue,ui,rnlis,rnap,nres,mk,akas, apid, nxres,res:text, kdf:hash_func, pkue,pkcp,
pkhss,pklis:public_key, skapws,skwls:symmetric_key,ue_ap_RNue, ue_ap_HNID,
ue_ap_NINNI,ue_ap_IMSI,ue_ap_RNAP,ue_ap_NRES,ap_ue_msg_auth_NRES,ap_ue_msg_auth_NINNI,ap_waaa_AT,ap_waaa_HNID,ap_waaa_NINNI,ap_waaa_IMS
I,ap_waaa_APID, waaa_ap_msg_auth_NINNI,ap_ue_RNlis,ap_ue_UI,ap_ue_RNAP, ap_waaa_NRES, waaa_ap msg_auth_NRES,ap_ue_AKAs,waaa_laaa_AT,
waaa_laaa_HNID, waaa_laaa_NINNI, waaa_laaa_IMSI, laaa_waaa_msg_auth_NINNI, waaa_ap RAND, waaa_ap PKue, waaa_ap UI,
waaa_ap_AKAs,waaa_ap_MK,laaa_hss_RNlis, laaa_hss_IMSI, hss_laaa_msg_auth_IMSI, laaa_waaa_NXRES, laaa_waaa_RNlis, laaa_waaa_PKue, laaa_waaa_UI,
laaa_hss_msg_auth_NINNI,hss_laaa_RNlis, hss_laaa_XRES, hss_laaa_PKue, hss_laaa_NINNI, hss_laaa_UI:protocol_id intruder_knowledge = {kdf, pkue,pkap,
pkhss,pklis} composition session1(ue,ap,waaa,laaa,hss,imsi,hnid,ts,at, ninni,res,rue,ui,rnlis,rmap,nres,mk,akas, apid, nxres,res,kdf,pkue,pkap, pkhss,pklis,skapws,skwls)
end role goal authentication_on ap_ue_msg_auth_NRES authentication_on ap_ue_msg_auth_NINNI authentication_on waaa_ap msg_auth_NINNI authentication_on
waaa_ap msg_auth_NRES authentication_on laaa_waaa_msg_auth_NINNI authentication_on hss_laaa_msg_auth_IMSI authentication_on laaa_hss_msg_auth_NINNI
secrecy_of ue_ap_RNue secrecy_of ue_ap_HNID secrecy_of ue_ap_AT secrecy_of ue_ap_NINNI secrecy_of ue_ap_IMSI secrecy_of ue_ap_RNAP secrecy_of
ue_ap_NRES secrecy_of ap_waaa_AT secrecy_of ap_waaa_HNID secrecy_of ap_waaa_NINNI secrecy_of ap_waaa_IMSI secrecy_of ap_waaa_APID secrecy_of
ap_ue_RNlis secrecy_of ap_ue_UI secrecy_of ap_ue_RNAP secrecy_of ap_waaa_NRES secrecy_of ap_ue_AKAs secrecy_of waaa_laaa_AT secrecy_of waaa_laaa_HNID
secrecy_of waaa_laaa_NINNI secrecy_of waaa_laaa_IMSI secrecy_of waaa_ap RAND secrecy_of waaa_ap PKue secrecy_of waaa_ap UI secrecy_of waaa_ap_AKAs
secrecy_of waaa_ap_MK secrecy_of laaa_hss_RNlis secrecy_of laaa_hss_IMSI secrecy_of laaa_waaa_NXRES secrecy_of laaa_waaa_RNlis secrecy_of laaa_waaa_PKue
secrecy_of laaa_waaa_UI secrecy_of hss_laaa_RNlis secrecy_of hss_laaa_XRES secrecy_of hss_laaa_PKue secrecy_of hss_laaa_NINNI secrecy_of hss_laaa_UI end goal
environment()
```


%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.02s, visitedNodes: 2 nodes, depth: 1 piles
%Statistics (ATSE mode): Analysed: 5 states, Reachable: 0 states, Translation: 0.05 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA4.2.2.1

```
role role UE (UE,AP,WAAA,LAAA,HSS: agent, IMSI,TS,HNID,AT,AMF,SQLN:text, LUIFunc,LNIGFunc,NKFunc,F1, F2, F3, F4, F5, KDF, AUTHFunc:hash_func, SK:symmetric_key,PKue,PKap : public_key, SND_AP,RCV_AP:channel(dy))played_by UE def=local State:nat,RNue,LUI,LNI,RNhss,NK, MAC,RES,CK, IK, AK,KASME,AUTN, RAND,MK,AKAs,RNAP,RNws: text const ue_ap_RNue, ue_ap_HNID, ue_ap_AT, ue_ap_LUI, ue_ap_IMSI, ue_ap_TS, ue_ap_RES, ap_ue_msg_auth_RES,ue_ap_msg_auth_LNI: protocol_id init State := 0 transition 1. State=0 ^ RCV_AP(start) => State:=1 ^ RNue:=new() ^ LUI:= LUIFunc(IMSI.SK.RNue) ^ SND_AP({HNID.AT.RNue.IMSI.LUI.TS}_PKap) ^ secret(RNue,ue_ap_RNue, {UE, AP}) ^ secret(HNID,ue_ap_HNID, {UE, AP}) ^ secret(AT,ue_ap_AT, {UE, AP}) ^ secret(LUI',ue_ap_LUI, {UE, AP}) ^ secret(IMSI,ue_ap_IMSI,{UE,AP}) ^ secret(TS,ue_ap_TS, {UE, AP}) 9. State=1 ^ RCV_AP({RAND'.AUTN'.RNhss'.{LNI'}_inv(PKue)})_PKue => State:=2 ^ LNI':= LNIGFunc(RNhss'.LUI.SK) ^ NK':= NKFunc(SK.xor(LUI, LNI')) ^ MAC':=F1(NK'.RAND'.AMF.SQLN) ^ RES':=F2(NK'.RAND') ^ CK':=F3(NK'.RAND') ^ IK':=F4(NK'.RAND') ^ AK':=F5(NK'.RAND') ^ KASME':=KDF(xor(SQLN, AK')).HNID.CK'.IK'.NK') ^ AUTN':=AUTHFunc(xor(SQLN, AK').AMF.MAC') ^ SND_AP({{RES'}_inv(PKue)})_PKap ^ secret(RES',ue_ap_RES, {UE, AP}) ^ witness (UE,AP, ap_ue_msg_auth_RES,RES') ^ request (UE,AP, ue_ap_msg_auth_LNI, LNI') 13. State=2 ^ RCV_AP({AKAs.RNws'}_PKue) => State:=3 ^ MK':= KDF(KASME.RNws') end role role _AP(UE,AP,WAAA,LAAA,HSS:agent,SKapws:symmetric_key,PKue,PKap:public_key,SND_UE,RCV_UE,SND_WAAA,RCV_WAAA:channel(dy))played_by AP def=local State:nat, RNue, HNID, AT, IMSI,LUI,TS,RNhss,RAND,AUTN,LNI,RNAP,RES,MK,AKAs,RNws:text const ap_waaa_AT,ap_waaa_HNID,ap_waaa_LUI,ap_waaa_IMSI,ap_waaa_RNue,ap_waaa_msg_auth_IMSI, ap_ue_RAND , ap_ue_AUTN, ap_ue_RNhss, ap_ue_LNI, ap_ue_msg_auth_LNI,ap_waaa_RES, ap_ue_msg_auth_RES,ap_ue_AKAs, ap_ue_RNws:protocol_id init State := 0 transition 2. State=0 ^ RCV_UE'({HNID.AT.RNue'.IMSI.LUI.TS}_PKap) => State:=1 ^ SND_WAAA'({AT.HNID.LUI'.RNue'.(IMSI)_inv(PKap)})_SKapws ^ secret(AT,ap_waaa_AT, {AP, WAAA}) ^ secret(HNID,ap_waaa_HNID, {AP, WAAA}) ^ secret(LUI',ap_waaa_LUI, {AP, WAAA}) ^ secret(IMSI,ap_waaa_IMSI, {AP, WAAA}) ^ secret(RNue',ap_waaa_RNue, {AP, WAAA}) ^ witness (AP, WAAA, waaa_ap_msg_auth_IMSI, IMSI) 8. State=1 ^ RCV_WAAA'({RNhss'.RAND'.AUTN'.LNI'.PKue')_SKapws) => State:=2 ^ SND_UE'({RAND'.AUTN'.RNhss'.{LNI'}_inv(PKap)})_PKue ^ secret(RAND', ap_ue_RAND, {AP, UE}) ^ secret(AUTN',ap_ue_AUTN,{AP,UE}) ^ secret(RNhss',ap_ue_RNhss, {AP,UE}) ^ secret(LNI',ap_ue_LNI,{AP,UE}) ^ witness(AP,UE,ue_ap_msg_auth_LNI,LNI') 10.State=2 ^ RCV_UE'({RES')_inv(PKue)})_PKap => State:=3 ^ SND_WAAA'({RES')_SKapws) ^ secret(RES',ap_waaa_RES, {AP, WAAA}) ^ request (AP,UE, ap_ue_msg_auth_RES,RES') 12. State=3 ^ RCV_UE'({{AKAs'.RNws'}_SKapws) => State:=4 ^ SND_UE'({AKAs.RNws'})_PKue ^ secret(AKAs, ap_ue_AKAs, {AP, UE}) ^ secret(RNws',ap_ue_RNws, {AP, UE}) end role role WAAA(UE,AP,WAAA, LAAA,HSS: agent, AKAs:text, KDF:hash_func,SKapws,SKwsls: symmetric_key,PKue,PKap,PKIs:public_key,SND_AP,RCV_AP,SND_LAAA,RCV_LAAA:channel(dy))played_by WAAA def=local State:nat, AT,HNID,LUI, RNue,IMSI, RAND,AUTN,KASME,RNhss,LNI,XRES,RNws,MK,RES:text const waaa_laia_AT,aaa_laia_HNID, waaa_laia_LUI, waaa_laia_RNue, waaa_laia_IMSI, waaa_ap_msg_auth_IMSI, waaa_ap_RNhss, waaa_ap_AUTN, waaa_ap_LNI,aaa_ap_PKue, waaa_ap_AKAs,aaa_ap_MK,aaa_ap_RNws:protocol_id init State := 0 transition 3. State=0 ^ RCV_AP'({AT.HNID. LUI'.RNue'.(IMSI)_inv(PKap)})_SKapws => State:=1 ^ SND_LAAA'({AT.HNID.LUI'.RNue'.IMSI')_SKwsls) ^ secret(AT,aaa_laia_AT,{WAAA,LAAA}) ^ secret(HNID,aaa_laia_HNID, {WAAA,LAAA}) ^ secret(LUI',aaa_laia_LUI, {WAAA,LAAA}) ^ secret(RNue',aaa_laia_RNue, {WAAA,LAAA}) ^ secret(IMSI,aaa_laia_IMSI, {WAAA,LAAA}) ^ request (WAAA,AP, waaa_ap_msg_auth_IMSI, IMSI) 7. State=1 ^ RCV_LAAA'({RNhss'.RAND'.AUTN'.KASME'.LNI'.PKue.XRES')_SKwsls) => State:=2 ^ SND_AP'({RNhss'.RAND'.AUTN'.LNI'.PKue')_SKapws) ^ secret(RNhss', waaa_ap_RNhss, {WAAA, AP}) ^ secret(RAND', waaa_ap_RAND, {WAAA, AP}) ^ secret(AUTN', waaa_ap_AUTN, {WAAA, AP}) ^ secret(LNI', waaa_ap_LNI, {WAAA, AP}) ^ secret(PKue', waaa_ap_PKue, {WAAA, AP}) 11. State=2 ^ RCV_AP'({RES')_SKapws) => State:=3 ^ RNws':=new() ^ MK':= KDF(KASME.RNws') ^ SND_AP'({AKAs.MK'.RNws'})_SKapws ^ secret(AKAs, waaa_ap_AKAs, {WAAA, AP}) ^ secret(MK', waaa_ap_MK, {WAAA, AP}) ^ secret(RNws', waaa_ap_RNws, {WAAA, AP}) end role role LAAA(UE,AP,WAAA, LAAA,HSS: agent, SKwsls:symmetric_key, PKue,PKhss,PKIs : public_key, SND_WAAA,RCV_WAAA,SND_HSS,RCV_HSS:channel(dy))played_by LAAA def=local State:nat, AT,HNID,LUI,RNue,RNhss,IMSI,RAND,AUTN,KASME,LNI,XRES:text const laia_hss_AT, laia_hss_RNue,laia_hss_LUI,laia_hss_IMSI,hss_laia_msg_auth_IMSI,laia_waaa_RNhss,laia_waaa_RAND,laia_waaa_AUTN,laia_waaa_KASME,laia_waaa_LNI,la aa_waaa_PKue, laia_waaa_XRES,laia_hss_msg_auth_XRES:protocol_id init State:=0 transition 4.State=0 ^ RCV_WAAA'({AT.HNID.LUI'.RNue'.IMSI')_SKwsls) => State:=1 ^ SND_HSS'({AT.RNue'. LUI'. (IMSI)_inv(PKIs)})_PKhss ^ secret(AT,laia_hss_AT, {LAAA, HSS}) ^ secret(RNue', laia_hss_RNue, {LAAA, HSS}) ^ secret(LUI',laia_hss_LUI, {LAAA, HSS}) ^ secret(IMSI,laia_hss_IMSI, {LAAA, HSS}) ^ witness (LAAA, HSS, hss_laia_msg_auth_IMSI, IMSI) 6. State=1 ^ RCV_HSS'({RNhss'.RAND'.AUTN'.KASME'.LNI'.PKue.{XRES'}_inv(PKhss)})_PKIs => State:=2 ^ SND_WAAA'({RNhss'.RAND'.AUTN'.KASME'.LNI'.PKue.XRES')_SKwsls) ^ secret(RNhss', laia_waaa_RNhss, {LAAA, WAAA}) ^ secret(RAND', laia_waaa_RAND, {LAAA, WAAA}) ^ secret(AUTN', laia_waaa_AUTN, {LAAA, WAAA}) ^ secret(LNI', laia_waaa_LNI, {LAAA, WAAA}) ^ secret(PKue', laia_waaa_PKue, {LAAA, WAAA}) ^ secret(XRES', laia_waaa_XRES, {LAAA, WAAA}) ^ request (LAAA,HSS, laia_hss_msg_auth_XRES, XRES) end role role HSS(UE,AP,WAAA, LAAA,HSS: agent, IMSI,HNID,AMF,SQLN:text, LUIFunc,LNIGFunc,NKFunc,F1, F2, F3, F4, F5, KDF, AUTHFunc:hash_func, SK:symmetric_key, PKhss,PKIs,Pkue : public_key, SND_LAAA,RCV_LAAA:channel(dy)) played_by HSS def= local State :nat, RNue,AT,LUI,LNI,RNhss,NK, MAC,XRES,CK, IK, AK,KASME,AUTN, RAND:text const hss_laia_RNhss, hss_laia_RAND,hss_laia_AUTN, hss_laia_KASME, hss_laia_LNI, hss_laia_PKue, hss_laia_XRES, hss_laia_msg_auth_IMSI, laia_hss_msg_auth_XRES:protocol_id init State := 0 transition 5. State=0 ^ RCV_LAAA'({AT.RNue'.LUI'.(IMSI)_inv(PKIs)})_PKhss) => State:=1 ^ RNhss':=new() ^ LUI':= LUIFunc(IMSI.SK.RNue) ^ LNI':= LNIGFunc(RNhss'.LUI'.SK) ^ NK':= NKFunc(SK.xor(LUI, LNI')) ^ RAND':=new() ^ MAC':=F1(NK'.RAND'.AMF.SQLN) ^ XRES':=F2(NK'.RAND') ^ CK':=F3(NK'.RAND') ^ IK':=F4(NK'.RAND') ^ AK':=F5(NK'.RAND') ^ KASME':=KDF(xor(SQLN, AK')).HNID.CK'.IK'.NK') ^ AUTN':= AUTHFunc (xor(SQLN,AK').AMF.MAC') ^ SND_LAAA'({RNhss'.RAND'. AUTN'. KASME'.LNI'.PKue'.{XRES')_inv(PKhss)})_PKIs ^ secret(RNhss',hss_laia_RNhss, {HSS,LAAA}) ^ secret(RAND',hss_laia_RAND, {HSS,LAAA}) ^ secret(AUTN',hss_laia_AUTN, {HSS,LAAA}) ^ secret( KASME', hss_laia_KASME, {HSS, LAAA}) ^ secret(LNI',hss_laia_LNI, {HSS, LAAA}) ^ secret(PKue',hss_laia_PKue, {HSS, LAAA}) ^ secret(XRES',hss_laia_XRES, {HSS, LAAA}) ^ request (HSS,LAAA, hss_laia_msg_auth_IMSI, IMSI) ^ witness (HSS,LAAA, laia_hss_msg_auth_XRES, XRES) end role role session1(UE,AP,WAAA, LAAA,HSS:agent, IMSI,TS,HNID,AT,AMF,SQLN,RNue,LUI,LNI,RNhss,NK, MAC,RES,CK,IK,AK,KASME, AUTN,RAND,MK,AKAs,RNAP,RNws,XRES :text,LUIFunc,LNIGFunc,NKFunc,F1,F2,F3,F4,F5,KDF, AUTHFunc:hash_func,Pkue,PKap,PKhss,PKIs:public_key,SK,SKapws, SKwsls:symmetric_key)def=local SND_AP1,RCV_AP1,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3, SND_WAAA4,RCV_WAAA4, SND_HSS4, RCV_HSS4, SND_LAAA5,RCV_LAAA5 : channel(dy) composition role _UE(UE,AP,WAAA, LAAA,HSS,IMSI,TS,HNID,AT,AMF,SQLN, LUIFunc,LNIGFunc,NKFunc,F1, F2, F3, F4, F5, KDF, AUTHFunc,SK,Pkue,PKap,SND_AP1,RCV_AP1) ^role _AP(UE,AP,WAAA, LAAA,HSS,SKapws,Pkue,PKap, SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2) ^role _WAAA(UE,AP,WAAA, LAAA,HSS, AKAs,KDF,SKapws,SKwsls,Pkue,PKap, PKIs,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3) ^role _LAAA(UE,AP,WAAA,LAAA,HSS,SKwsls,Pkue,PKhss,PKIs, SND_WAAA4,RCV_WAAA4, SND_HSS4, RCV_HSS4) ^role _HSS(UE,AP,WAAA, LAAA,HSS,IMSI,HNID,AMF,SQLN,LUIFunc,LNIGFunc,NKFunc,F1, F2, F3, F4, F5, KDF, AUTHFunc, SK,PKhss,PKIs, PKue _SND_LAAA5,RCV_LAAA5) end role environment() def=const ue:agent,ap:agent,aaa:agent,laia:agent,hss:agent, imsi,ts,hnid,at,amf,sqln,mue,lui,lni,rhss,nk, mac,res,ck, ik, ak,kasme,aun, rand,mk,akas,rnap,rnws,xres:text, luifunc,lnigfunc,nkfunc,f1, f2, f3, f4, f5, kdf, authfunc:hash_func, pkue,pkap, pkhss,pkls:public_key, sk,skapws,skwsls:symmetric_key, ue_ap_RNue, ue_ap_HNID, ue_ap_AT, ue_ap_LUI, ue_ap_IMSI, ue_ap_TS, ue_ap_RES, ap_waaa_AT, ap_waaa_HNID, ap_waaa_LUI, ap_waaa_IMSI, ap_waaa_RNue, ap_ue_RAND, ap_ue_AUTN, ap_ue_RNhss, ap_ue_LNI, ap_waaa_RES,ap_ue_AKAs, ap_ue_RNws,aaa_laia_AT,aaa_laia_HNID, waaa_laia_LUI, waaa_laia_RNue, waaa_laia_IMSI, waaa_ap_RAND, waaa_ap_AUTN, waaa_ap_LNI, waaa_ap_PKue, waaa_ap_AKAs, waaa_ap_MK, waaa_ap_RNws, laia_hss_AT, laia_hss_RNue, laia_hss_LUI, laia_hss_IMSI, laia_waaa_RNhss, laia_waaa_RAND, laia_waaa_AUTN,laia_waaa_KASME, laia_waaa_LNI,laia_waaa_PKue,laia_waaa_XRES, hss_laia_RNhss, hss_laia_RAND,hss_laia_AUTN, hss_laia_KASME, hss_laia_LNI, hss_laia_PKue, hss_laia_XRES, waaa_ap_msg_auth_IMSI, ue_ap_msg_auth_LNI,ap_ue_msg_auth_RES, hss_laia_msg_auth_IMSI, laia_hss_msg_auth_XRES:protocol_id intruder_knowledge = {luifunc,lnigfunc,nkfunc,f1, f2, f3, f4, f5, kdf, authfunc,pkue,pkap, pkhss,pkls} composition session1(ue,ap,aaa,laia,hss,imsi,ts,hnid,at,amf,sqln,mue,lui,lni,rhss,nk, mac,res,ck, ik, ak,kasme,aun,rand,mk,akas,rnap,rnws,xres, luifunc,lnigfunc,nkfunc,f1, f2, f3, f4, f5, kdf, authfunc,pkue,pkap, pkhss,pkls,skapws,skwsls) end role goal authentication on laia_hss_msg_auth_XRES authentication on hss_laia_msg_auth_IMSI authentication on ap_ue_msg_auth_RES authentication on ue_ap_msg_auth_LNI authentication on waaa_ap_msg_auth_IMSI secrecy of hss_laia_XRES secrecy of hss_laia_LNI secrecy of hss_laia_KASME secrecy of hss_laia_AUTN secrecy of hss_laia_RAND secrecy of hss_laia_RNhss secrecy of laia_waaa_AUTN secrecy of laia_waaa_LNI secrecy of laia_waaa_RAND secrecy of laia_waaa_RNhss secrecy of laia_hss_IMSI secrecy of laia_waaa_KASME secrecy of laia_waaa_PKue secrecy of laia_waaa_XRES secrecy of laia_hss_LUI secrecy of laia_hss_RNue secrecy of laia_hss_AT secrecy of ap_ue_RNhss secrecy of waaa_ap_PKue secrecy of waaa_ap_AKAs secrecy of waaa_ap_MK secrecy of waaa_ap_RNws secrecy of waaa_ap_LNI secrecy of ue_ap_IMSI secrecy of waaa_ap_AUTN secrecy of ue_ap_AT secrecy of ue_ap_TS secrecy of ap_waaa_LUI secrecy of ue_ap_RNue secrecy of ue_ap_HNID secrecy of ue_ap_LUI secrecy of ue_ap_RES secrecy of ap_waaa_AT secrecy of ap_waaa_HNID secrecy of ap_waaa_IMSI secrecy of ap_waaa_RNue secrecy of ap_ue_RAND secrecy of ap_ue_AUTN secrecy of ap_ue_LNI secrecy of ap_waaa_RES secrecy of ap_ue_AKAs secrecy of ap_ue_RNws secrecy of waaa_laia_AT secrecy of waaa_laia_HNID secrecy of waaa_laia_LUI secrecy of waaa_laia_RNue secrecy of waaa_laia_IMSI secrecy of waaa_ap_RAND secrecy of waaa_ap_RNhss end goal environment()
```

%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.06s, visitedNodes: 2 nodes, depth: 1 piles

%Statistics (ATSE mode): Analysed: 5 states, Reachable: 0 states, Translation: 0.10 seconds, Computation: 0.00 seconds

%% PROTOCOL: REPS-AKA4.2.2.2

```
role role UE (UE,AP,WAAA, LAAA: agent, IMSI,HNID,AT,PLUI,RES,AUTN,KASME,LNI:text, LUIFunc,UKFunc,KDF:hash_func, PKue,PKap : public_key, SND_AP,RCV_AP:channel(dy))played_by UE def=local State:nat,RNue,LUI,UK,RNIs,RNws,NRES,NKASME,NAUTN,NLNI,MK,AKAs: text const ue_ap_RNue,ue_ap_HNID,ue_ap_AT,ue_ap_LUI, ue_ap_IMSI, ue_ap_RES, ap_ue_msg_auth_NRES: protocol_id init State := 0 transition 1. State=0 ^ RCV_AP(start) => State:=1 ^ RNue':=new() ^ LUI':= LUIFunc(IMSI.PLUI.RNue') ^ SND_AP'({HNID.AT.RNue'.IMSI.LUI'})_PKap) ^ secret(RNue,ue_ap_RNue, {UE, AP}) ^ secret(HNID,ue_ap_HNID, {UE, AP}) ^ secret(AT,ue_ap_AT, {UE, AP}) ^ secret(LUI',ue_ap_LUI, {UE, AP}) ^ secret(IMSI,ue_ap_IMSI, {UE, AP}) 9. State=1 ^ RCV_AP'({RNIs'.RNws'.NAUTN'.NLNI'})_PKue) => State:=2 ^ UK':= UKFunc(LUI'.RNIs'.RNws') ^ NRES':= KDF(RES.UK') ^ AUTN':= KDF(AUTN.UK') ^ NKASME':= KDF(KASME.UK') ^ NLNI':= KDF(LNI.UK') ^ SND_AP'({NRES')_inv(PKue)})_PKap ^ secret(NRES',ue_ap_RES, {UE, AP}) ^ witness (UE,AP, ap_ue_msg_auth_NRES,NRES') 13. State=2 ^ RCV_AP'({AKAs}_PKue) => State:=3 ^ MK':= KDF(NKASME.RNws') end role role AP(UE,AP,WAAA, LAAA: agent, SKapws:symmetric_key,PKue,PKap : public_key, SND_UE,RCV_UE,SND_WAAA,RCV_WAAA:channel(dy))played_by AP def=local
```

```

State:nat,RNue,HNID,AT,IMSI,LUI,RNws,RNls,NAUTN,NLNI,NRES,MK,AKAs:text const ap_waaa_AT,ap_waaa_HNID,ap_waaa_LUI,
ap_waaa_IMSI,ap_waaa_RNue, waaa_ap_msg_auth_IMSI, ap_ue_RNls, ap_ue_RNws, ap_ue_NAUTN,ap_ue_NLNI, ap_waaa_NRES, ap_ue_msg_auth_NRES,
ap_ue_AKAs:protocol_id init State := 0 transition 2. State=0 ^ RCV_UE({HNID,AT,RNue,IMSI,LUI}_PKap) => State:=1 ^
SND_WAAA({AT,HNID,LUI,RNue'}_IMSI)_inv(PKAp)_SKapws ^ secret(AT,ap_waaa_AT, {AP, WAAA})^ secret(HNID,ap_waaa_HNID, {AP, WAAA})^
secret(LUI,ap_waaa_LUI, {AP, WAAA})^ secret(IMSI,ap_waaa_IMSI, {AP, WAAA})^ secret(RNue,ap_waaa_RNue, {AP, WAAA})^ witness (AP, WAAA,
aaa_ap_msg_auth_IMSI, IMSI) 8. State=1 ^ RCV_WAAA({RNls',RNws,NAUTN',NLNI'}_SKapws) => State:=2 ^SND_UE({RNls',RNws,NAUTN',NLNI'}_PKue)
^ secret(RNls',ap_ue_RNls, {AP, UE})^ secret(RNws,ap_ue_RNws, {AP, UE})^ secret(NAUTN',ap_ue_NAUTN, {AP, UE})^ secret(NLNI',ap_ue_NLNI, {AP, UE})
10. State=2 ^ RCV_UE({NRES'}_inv(PKue))_PKap => State:=3 ^ SND_WAAA({NRES'}_SKapws) ^ secret(NRES',ap_waaa_NRES, {AP, WAAA})^ request
(AP,UE, ap_ue_msg_auth_NRES,NRES) 12. State=3 ^ RCV_WAAA({AKAs,MK'}_SKapws) => State:=4^SND_UE({AKAs}_PKue) ^ secret(AKAs,ap_ue_AKAs,
{AP, UE})end role role role_WAAA(UE,AP,WAAA, LAAA: agent, AKAs:text, KDF:hash_func, SKapws,SKwls:symmetric_key, PKap:
public_key,SND_AP,RCV_AP,SND_LAAA,RCV_LAAA:channel(dy))played_by WAAA def=local State:nat,
AT,HNID,LUI,RNue,IMSI,MK,RNls,RNws,NAUTN,NKASME,NLNI,NXRES,NRES:text const waaa_laia_AT, waaa_laia_HNID, waaa_laia_LUI, waaa_laia_RNue,
aaa_laia_IMSI, waaa_laia_RNws, waaa_ap_msg_auth_IMSI, waaa_ap_RNls, waaa_ap_RNws, waaa_ap_NAUTN, waaa_ap_NLNI,
aaa_ap_AKAs,aaa_ap_MK:protocol_id init State := 0 transition 3. State=0 ^ RCV_AP({AT,HNID,LUI,RNue'}_IMSI)_inv(PKAp)_SKapws) => State:=1^
RNws:=new()^ SND_LAAA({AT,HNID,LUI,RNue'.IMSI,RNws'}_SKwls) ^ secret(AT,aaa_laia_AT, {WAAA,LAAA})^ secret(HNID,aaa_laia_HNID,
{WAAA,LAAA})^ secret(LUI,aaa_laia_LUI, {WAAA,LAAA})^ secret(RNue',aaa_laia_RNue, {WAAA,LAAA})^ secret(IMSI,aaa_laia_IMSI,
{WAAA,LAAA})^ secret(RNws',aaa_laia_RNws, {WAAA,LAAA})^ request (WAAA,AP, waaa_ap_msg_auth_IMSI, IMSI)7. State=1 ^
RCV_LAAA({RNls',NAUTN',NKASME',NXRES',NLNI'}_SKwls) => State:=2^SND_AP({RNls',RNws,NAUTN',NLNI'}_SKapws) ^ secret(RNls', waaa_ap_RNls,
{WAAA, AP})^ secret(RNws, waaa_ap_RNws, {WAAA, AP})^ secret(NAUTN', waaa_ap_NAUTN, {WAAA, AP})^ secret(NLNI', waaa_ap_NLNI, {WAAA, AP})11.
State=2 ^ RCV_AP({NRES'}_SKapws) => State:=3 ^ MK':= KDF(NKASME,RNws) ^ SND_AP({AKAs,MK'}_SKapws)^ secret(AKAs, waaa_ap_AKAs, {WAAA,
AP})^ secret(MK', waaa_ap_MK, {WAAA, AP})end role role role_LAAA(UE,AP,WAAA, LAAA: agent, IMSI,PLUI,XRES,AUTN,KASME,LNI:text,
LUIFunc,UKFunc,KDF:hash_func, SKwls:symmetric_key,SND_WAAA,RCV_WAAA:channel(dy))played_by LAAA def=local State:nat, AT,HNID,LUI,
RNue,RNws,RNls,UK,NXRES,NAUTN,NKASME,NLNI:text const laia_waaa_RNls,laia_waaa_NAUTN, laia_waaa_NKASME,
laia_waaa_NXRES,laia_waaa_NLNI:protocol_id init State := 0 transition 4. State=0 ^ RCV_WAAA({AT,HNID,LUI,RNue'.IMSI,RNws'}_SKwls) => State:=1 ^
RNls':=new()^ LUI':= LUIFunc(IMSI,PLUI,RNue') ^ UK':= UKFunc(LUI',RNls',RNws') ^ NXRES':= KDF(XRES,UK) ^ NAUTN':= KDF(AUTN,UK) ^ NKASME':=
KDF(KASME,UK) ^ NLNI':= KDF(LNI,UK) ^ SND_WAAA({RNls',NAUTN',NKASME',NXRES',NLNI'}_SKwls) ^ secret(RNls',laia_waaa_RNls, {LAAA,
WAAA})^secret(NAUTN',laia_waaa_NAUTN,{LAAA,WAAA})^secret(NKASME',laia_waaa_NKASME,{LAAA,WAAA})^secret(NXRES',laia_waaa_NXRES,{L
AAA,WAAA})^secret(NLNI', laia_waaa_NLNI,{LAAA,WAAA}) end role session1(UE,AP,WAAA,LAAA:agent,IMSI,
HNID,AT,PLUI,RES,AUTN,KASME,LNI,RNue,LUI,UK,RNls,RNws,NRES,NKASME, NAUTN,NLNI,MK,AKAs,XRES,NXRES:text,LUIFunc,UKFunc,
KDF:hash_func,PKue,PKap:public_key,SKapws,SKwls:symmetric_key)def=local SND_AP1,RCV_AP1,SND_UE2, RCV_UE2, SND_WAAA2,
RCV_WAAA2,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3,SND_WAAA4,RCV_WAAA4:channel(dy)composition role_UE(UE,AP,WAAA,
LAAA,IMSI,HNID,AT, PLUI,RES,AUTN,KASME, LNI,LUIFunc, UKFunc,KDF,PKue,PKap,SND_AP1, RCV_AP1)/role_AP
(UE,AP,WAAA,LAAA,SKapws,PKue,PKap,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2)/role_WAAA(UE,AP,WAAA,LAAA,AKAs,KDF,
SKapws,SKwls,PKap,SND_AP3,RCV_AP3,SND_LAAA3, RCV_LAAA3)/role_LAAA(UE,AP,WAAA,LAAA,IMSI,PLUI ,XRES,AUTN,KASME,LNI,
LUIFunc,UKFunc,KDF,SKwls, SND_WAAA4,RCV_WAAA4) end role role environment() def=const ue:agent,ap:agent,aaa
:agent,laia:agent,hss:agent,imsi,hnid,at,plui,res,autn,kasme,lni,rnue,lui,uk,rnls,rnws,nres,nkasme,nautn,nlni,mk,akas ,xres,nxres: text,luifunc,ukfunc,kdf:
hash_func,pkue,pkap:public_key,skapws,skwls:symmetric_key,ue_ap_RNue,ue_ap_HNID,ue_ap_AT,ue_ap_LUI,ue_ap_IMSI,ue_ap_RES,ap_waaa_AT,
ap_waaa_HNID, ap_waaa_LUI, ap_waaa_IMSI, ap_waaa_RNue,ap_ue_RNls, ap_ue_RNws, ap_ue_NAUTN,ap_ue_NLNI, ap_waaa_NRES,
ap_ue_AKAs,aaa_waaa_laia_AT, waaa_laia_HNID, waaa_laia_LUI, waaa_laia_RNue, waaa_laia_IMSI,aaa_waaa_RNws,aaa_ap_RNls, waaa_ap_RNws,
aaa_ap_NAUTN, waaa_ap_NLNI, waaa_ap_AKAs,aaa_ap_MK,laia_waaa_RNls,laia_waaa_NAUTN, laia_waaa_NKASME, laia_waaa_NXRES,
laia_waaa_NLNI,ap_ue_msg_auth_NRES, waaa_ap_msg_auth_IMSI:protocol_id intruder_knowledge = {luifunc,ukfunc,kdf,pkue,pkap}composition
session1(ue,ap,aaa,laia,imsi,hnid,at,plui,res,autn,kasme,lni, rnue,lui,uk,rnls,rnws,nres,nkasme,nautn,nlni,mk,akas,xres,
nxres,luifunc,ukfunc,kdf, pkue,pkap,skapws,skwls)end role goal authentication_on waaa_ap_msg_auth_IMSI authentication_on ap_ue_msg_auth_NRES secrecy_of ue_ap_RNue secrecy_of
ue_ap_HNID secrecy_of ue_ap_AT secrecy_of ue_ap_LUI secrecy_of ue_ap_IMSI secrecy_of ue_ap_RES secrecy_of ap_waaa_AT secrecy_of ap_waaa_HNID
secrecy_of ap_waaa_LUI secrecy_of ap_waaa_IMSI secrecy_of ap_waaa_RNue secrecy_of ap_ue_RNls secrecy_of ap_ue_RNws secrecy_of ap_ue_NAUTN
secrecy_of ap_ue_NLNI secrecy_of ap_waaa_NRES secrecy_of ap_ue_AKAs secrecy_of waaa_laia_AT secrecy_of waaa_laia_HNID secrecy_of waaa_laia_LUI
secrecy_of waaa_laia_RNue secrecy_of waaa_laia_IMSI secrecy_of waaa_laia_RNws secrecy_of waaa_ap_RNls secrecy_of waaa_ap_RNws secrecy_of
aaa_ap_NAUTN secrecy_of waaa_ap_NLNI secrecy_of waaa_ap_AKAs secrecy_of waaa_ap_MK secrecy_of laia_waaa_RNls secrecy_of laia_waaa_NAUTN
secrecy_of laia_waaa_NKASME secrecy_of laia_waaa_NXRES secrecy_of laia_waaa_NLNI end goal environment()
%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.02s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 3 states, Reachable: 0 states, Translation: 0.04 seconds, Computation: 0.00 seconds
%% % PROTOCOL: REPS-AKA4.3.1
role role_UE (UE,AP,WAAA: agent, IMSI,AT, NINNI,LAPID,RES,UI:text,KDF:hash_func, PKue,PKap : public_key, SND_AP,RCV_AP:channel(dy)) played by UE
def=local State:nat, RNue,RNap,RNws,NRES,AKAs,MK: text const ue_ap_RNue, ue_ap_LAPID, ue_ap_NINNI,ue_ap_IMSI, ue_ap_msg_auth_NINNI,
aaa_ue_msg_auth_NINNI,ue_ap_NRES,ue_ap_RNue2,ue_ap_RNue,ue_ap_msg_auth_NRES,aaa_ue_msg_auth_NRES,ap_ue_auth_RNap:protocol_id init State:=
0 transition 1. State=0 ^ RCV_AP(start) => State:=1^ RNue':=new()^ SND_AP({RNue',AT,LAPID,{NINNI}_inv(PKue)_IMSI)_PKap) ^ secret(RNue',ue_ap_RNue,
{UE, AP})^ secret(LAPID ,ue_ap_LAPID, {UE, AP})^ secret(NINNI ,ue_ap_NINNI, {UE, AP}) ^ secret(IMSI ,ue_ap_IMSI, {UE, AP}) ^ witness (UE,AP,
ap_ue_msg_auth_NINNI,NINNI)^ witness (UE,WAAA, waaa_ue_msg_auth_NINNI, NINNI)9. State=1 ^ RCV_AP({RNap',RNws'}_PKue) => State:=2 ^
RNue':=new()^ NRES':= KDF(RES,RNue'.RNws') ^ SND_AP({RNue'.RNws'}_NRES')^ secret(RNue',ue_ap_NRES, {UE, AP})^
secret(RNue',ue_ap_RNue2, {UE, AP})^ secret(NRES',ue_ap_NRES, {UE, AP})^ witness (UE,AP, ap_ue_msg_auth_NRES,NRES') ^ witness (UE,WAAA,
aaa_ue_msg_auth_NRES,NRES') ^ witness (UE,AP, ap_ue_auth_RNap,RNap) 13. State=2 ^RCV_AP({AKAs}_PKue)=>State:=3^MK':=KDF(NINNI,UI.NRES)end
role role_role_UE(UE,AP,WAAA:agent,SKapws:symmetric_key,PKue,PKap:public_key, SND_UE,RCV_UE, SND_WAAA,RCV_WAAA:channel(dy))played by AP
def=local State:nat,LAPID,RNue,AT,NINNI,IMSI,RNap,RNws,NRES,MK,AKAs:text const ap_waaa_AT, ap_waaa_LAPID, ap_waaa_NINNI,
ap_waaa_IMSI,ap_waaa_RNap,ap_ue_msg_auth_NINNI,ap_ue_RNap,ap_ue_RNws,ap_waaa_auth_RNap,ap_waaa_NRES,ap_ue_msg_auth_NRES,ap_ue_auth_RNap,
aaa_ap_msg_auth_NRES,ap_ue_AKAs:protocol_id init State := 0 transition 2. State=0 ^RCV_UE({RNue'.AT,
LAPID,{NINNI}_inv(PKue)_IMSI)_PKap)=>State:=1^RNap':=new()^ SND_WAAA ({AT,{NINNI}_inv(PKue)_IMSI,LAPID,RNap')_SKapws) ^
secret(AT,ap_waaa_AT, {AP, WAAA})^ secret(LAPID,ap_waaa_LAPID, {AP, WAAA})^ secret(NINNI,ap_waaa_NINNI, {AP, WAAA})^ secret(IMSI,ap_waaa_IMSI,
{AP, WAAA})^ secret(RNap',ap_waaa_RNap, {AP, WAAA})^ request (AP,UE, ap_ue_msg_auth_NINNI,NINNI) 8. State=1 ^
RCV_WAAA({RNap',RNws'}_PKue)_SKapws) => State:=2 ^SND_UE({RNap',RNws'}_PKue) ^ secret(RNap',ap_ue_RNap, {AP, UE})^ secret(RNws',ap_ue_RNws,
{AP, UE})^ request (AP,WAAA, ap_waaa_auth_RNap, RNap')10. State=2 ^ RCV_UE({RNue'.RNap'}_NRES')_inv(PKue)_PKap) => State:=3 ^
SND_WAAA({RNue'.NRES'}_inv(PKue))_SKapws) ^ secret(NRES',ap_waaa_NRES, {AP, WAAA}) ^ request (AP,UE,ap_ue_msg_auth_NRES,NRES') ^ request
(AP,UE, ap_ue_auth_RNap,RNap')^ witness (AP,WAAA, waaa_ap_msg_auth_NRES,NRES') 12. State=3 ^ RCV_WAAA({MK',AKAs}_SKapws) => State:=4 ^
SND_UE({AKAs}_PKue) ^ secret(AKAs ,ap_ue_AKAs, {AP, UE})end role role_role_WAAA(UE,AP,WAAA: agent,
AKAs,XRES,NINNI,UI:text,KDF:hash_func,SKapws:symmetric_key, PKue,PKap: public_key,SND_AP,RCV_AP:channel(dy))played_by WAAA def=local
State:nat,AT,IMSI,LAPID,RNap,RNws,NRES,RNue,NXRES,MK:text const waaa_ap_RNap, waaa_ap_RNws, waaa_ap_PKue, waaa_ue_msg_auth_NINNI,
ap_waaa_auth_RNap, waaa_ap_MK, waaa_ap_AKAs, waaa_ue_msg_auth_NRES:protocol_id init State := 0 transition 3. State=0 ^
RCV_AP({AT,{NINNI}_inv(PKue)_IMSI,LAPID,RNap')_SKapws) => State:=1^ RNws:=new()^ SND_AP({RNap',RNws'}_PKue)_SKapws) ^
secret(RNap',ap_waaa_RNap, {WAAA,AP})^ secret(RNws',aaa_ap_RNws, {WAAA,AP})^ secret(PKue,aaa_ap_PKue, {WAAA,AP})^ request (WAAA,UE,
aaa_ue_msg_auth_NINNI, NINNI) ^ witness (WAAA,AP, ap_waaa_auth_RNap, RNap')7. State=1 ^ RCV_AP({RNue'.NRES'}_inv(PKue))_SKapws) => State:=2 ^
NXRES':= KDF(XRES,RNue'.RNws') ^ MK':= KDF(NINNI,UI.NXRES')^SND_AP({MK',AKAs}_SKapws)^ secret(MK', waaa_ap_MK, {WAAA, AP})^ secret(AKAs,
aaa_ap_AKAs, {WAAA, AP})^ witness (WAAA,UE, waaa_ue_msg_auth_NRES,NRES')end role role_session1(UE,AP,WAAA:agent,IMSI,AT,
NINNI,LAPID,RES,UI, RNue,RNap,RNws,NRES,AKAs,MK,XRES,NXRES:text, KDF:hash_func, PKue,PKap: public_key,SKapws:symmetric_key) def=local
SND_AP1,RCV_AP1, SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2, SND_AP3,RCV_AP3: channel(dy) composition role_UE(UE,AP,WAAA,IMSI,AT,
NINNI,LAPID,RES,UI,KDF,PKue,PKap,SND_AP1,RCV_AP1) /role_AP(UE,AP,WAAA,SKapws,PKue,PKap,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2)
/role_WAAA(UE,AP,WAAA, AKAs,XRES,NINNI,UI,KDF,SKapws,PKue,PKap,SND_AP3,RCV_AP3) end role role_environment() def= const
ue:agent,ap:agent,aaa: agent,imsi,at,ninni,lapid,res,ui,rnue,rnap,rnws,nres,akas,mk,xres,nxres:text,kdf:hash_func,pkue,pkap:public_key,skapws: symmetric_key,
ue_ap_RNue,ue_ap_LAPID,ue_ap_NINNI,ue_ap_IMSI,ue_ap_msg_auth_NINNI,aaa_ue_msg_auth_NINNI,ue_ap_NRES,ue_ap_RNue2,ue_ap_NRES,
ap_ue_msg_auth_NRES, waaa_ue_msg_auth_NRES,ap_ue_auth_RNap,ap_waaa_AT,ap_waaa_LAPID,ap_waaa_NINNI, ap_waaa_IMSI,ap_waaa_RNap,
ap_ue_RNap,ap_ue_RNws,ap_waaa_auth_RNap, ap_waaa_NRES, waaa_ap_msg_auth_NRES,ap_ue_AKAs,aaa_ap_RNap, waaa_ap_RNws, waaa_ap_PKue,
aaa_ap_MK, waaa_ap_AKAs:protocol_id intruder_knowledge = {kdf, pkue,pkap}composition session1(ue,ap,aaa,imsi,at, ninni,lapid,res,ui,
rnue,rnap,rnws,nres,akas,mk,xres,nxres,kdf,pkue,pkap, skapws)end role goal authentication_on ap_ue_msg_auth_NINNI authentication_on waaa_ue_msg_auth_NINNI
authentication_on ap_ue_auth_NRES authentication_on waaa_ue_auth_RNap authentication_on ap_waaa_auth_RNap authentication_on waaa_ap_msg_auth_NRES
secrecy_of ue_ap_RNue secrecy_of ue_ap_NRES secrecy_of ap_waaa_AT secrecy_of ap_waaa_LAPID secrecy_of ue_ap_NINNI secrecy_of ue_ap_IMSI secrecy_of
ue_ap_NRES secrecy_of ap_waaa_RNap secrecy_of ap_ue_RNap secrecy_of ap_ue_RNws secrecy_of ap_waaa_NRES secrecy_of ap_ue_AKAs secrecy_of
aaa_ap_RNap secrecy_of waaa_ap_RNws secrecy_of waaa_ap_PKue secrecy_of waaa_ap_MK secrecy_of waaa_ap_AKAs end goal environment()
%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.02s, visitedNodes: 3 nodes, depth: 2 piles
%%Statistics (ATSE mode): Analysed: 5 states, Reachable: 1 states, Translation: 0.02 seconds, Computation: 0.00 seconds
%% % PROTOCOL: EPS-AKA4.3.2

```

```

role role UE (UE,AP,WAAA: agent, IMSI,LUI,AT,LAPID,LNINLNI,KASME,RES:text,NUIFunc,UNIDFunc,KDF:hash_func, PKue,PKap : public_key,
SND_AP,RCV_AP:channel(dy))played_by UE def=local State:nat,RNue,NUI,RNws,RNap,UNID,NKASME,NRES,NMK,AKAs: text const ue_ap_RNue, ue_ap_NUI,
ue_ap_AT, ue_ap_LAPID, ue_ap_IMSI, ue_ap_NRES, ap_ue_msg_auth_NRES: protocol_id init State := 0 transition 1. State=0 / RCV_AP(start) => State:=1 /
RNue' := new() / NUI' := NUIFunc(LUI,RNue) / SND_AP({AT,RNue'.IMSI,NUI'.LAPID}_PKap) / secret(RNue'.ue_ap_RNue, {UE, AP}) / secret(NUI', ue_ap_NUI,
{UE, AP}) / secret(AT, ue_ap_AT, {UE, AP}) / secret(LAPID, ue_ap_LAPID, {UE, AP}) / secret(IMSI, ue_ap_IMSI, {UE, AP}) 9. State=1 /
RCV_AP({RNws'.UNID'}_PKue) => State:=2 / UNID' := UNIDFunc(LNINLNI,RNws') / NKASME' := KDF(KASME.RNws') / NRES' := KDF(RES.RNws'.RNue) /
SND_AP({NRES'}.inv(PKue))_PKap / secret(NRES',ue_ap_NRES, {UE, AP}) / witness (UE,AP, ap_ue_msg_auth_NRES,NRES) 13. State=2 /
RCV_AP({AKAs.RNap}_PKue) => State:=3 / NMK' := KDF(NKASME.RNap) end role role role AP(UE,AP,WAAA: agent,
SKapws:symmetric_key,PKue,PKap:public_key,SND,UE,RCV,UE,SND_WAAA,RCV_WAAA:channel(dy))played_by AP def=local State:nat,
RNue,AT,IMSI,NUI,LAPID,RNap,RNws,UNID, NRES,NMK, AKAs:text const ap_waaa_AT,ap_waaa_RNue, ap_waaa_IMSI,ap_waaa_LAPID, ap_waaa_NUI,
ap_waaa_msg_auth_NUI, ap_ue_RNws, ap_ue_UNID,ap_waaa_NRES, ap_waaa_RNap, ap_ue_msg_auth_NRES,
ap_ue_msg_auth_NRES,ap_ue_AKAs,ap_ue_RNap:protocol_id init State := 0 transition 2. State=0 / RCV_UE({AT,RNue'.IMSI,NUI'.LAPID}_PKap) => State:=1 /
SND_WAAA({AT,RNue'.IMSI,LAPID,{NUI'}_inv(PKap))_SKapws) / secret(AT,ap_waaa_AT, {AP, WAAA}) / secret(RNue',ap_waaa_RNue, {AP, WAAA}) /
secret(IMSI,ap_waaa_IMSI, {AP, WAAA}) / secret(LAPID,ap_waaa_LAPID, {AP, WAAA}) / secret(NUI,ap_waaa_NUI, {AP, WAAA}) / witness
(AP,WAAA,ap_waaa_msg_auth_NUI, NUI') 8. State=1 / RCV_WAAA({RNws'.PKue.UNID'}_SKapws) => State:=2 / SND_UE({RNws'.UNID'}_PKue) /
secret(RNws', ap_ue_RNws, {AP, UE}) / secret(UNID', ap_ue_UNID, {AP, UE}) 10. State=2 / RCV_UE({NRES'}.inv(PKue))_PKap => State:=3 /
RNap' := new() / SND_WAAA({RNap'}.NRES'}.inv(PKap))_SKapws) / secret(NRES',ap_waaa_NRES, {AP,WAAA}) / secret(RNap',ap_waaa_RNap, {AP,WAAA}) / request
(AT,UE,ap_ue_msg_auth_NRES,NRES) / witness (AP,WAAA, ap_waaa_msg_auth_NRES,NRES) 12. State=3 / RCV_WAAA({NMK'.AKAs}_SKapws) => State:=4 /
SND_UE({AKAs.RNap}_PKue) / secret(AKAs, ap_ue_AKAs, {AP, UE}) / secret(RNap, ap_ue_RNap, {AP, UE}) end role role role WAAA(UE,AP,WAAA: agent,
LUI,AKAs,IMSI,LNINLNI,XRES,KASME:text, NUIFunc,UNIDFunc,KDF:hash_func, SKapws:symmetric_key, PKue,PKap:
public_key,SND,AP,RCV_AP:channel(dy))played_by WAAA def=local State:nat,AT,RNue,UNID,NKASME,NMK,RNap,LAPID,NRES:text const
aaaa_ap_RNws, aaaa_ap_PKue,aaaa_ap_UNID,aaaa_ap_NMK,aaaa_ap_AKAs,aaaa_ap_msg_auth_NRES,aaaa_ap_msg_auth_NUI:protocol_id init State:= 0
transition 3.State=0 /
RCV_AP
({AT,RNue'.IMSI,LAPID,{NUI'}_inv(PKap))_SKapws) => State:=1 / RNws' := new() / NUI' := NUIFunc(LUI,RNue') / UNID' := UNIDFunc(LNINLNI,RNws') / SND_AP({
RNws'.PKue.UNID'}_SKapws) / secret (RNws',aaaa_ap_RNws, {WAAA,AP}) / secret(PKue,aaaa_ap_PKue, {UNID,WAAA,AP}) / secret(UNID,aaaa_ap_UNID,
{WAAA,AP}) / request (WAAA,AP, aaaa_ap_msg_auth_NUI, NUI') 7. State=1 / RCV_AP({RNap'}.NRES'}.inv(PKap))_SKapws) => State:=2 / NKASME' :=
KDF(KASME.RNws') / NXRES' := KDF(XRES.RNws'.RNue) / NMK' := KDF(NKASME.RNap') / SND_AP({NMK'.AKAs}_SKapws) / secret(NMK', aaaa_ap_NMK,
{WAAA, AP}) / secret(AKAs, aaaa_ap_AKAs, {WAAA, AP}) / request(WAAA,AP, aaaa_ap_msg_auth_NRES,NRES) end role role session1(UE,AP,WAAA:agent,
IMSI,LUI,AT,LAPID,LNINLNI,KASME,RES,RNue,NUI,RNws,RNap,UNID,NKASME,NRES,NMK,AKAs,XRES,NXRES:text,NUIFunc,UNIDFunc,KDF:hash_func,
PKue,PKap:public_key,SKapws:
SND,AP1,RCV,AP1,SND,UE2,RCV,UE2,SND_WAAA2,RCV_WAAA2,SND,AP3,RCV,AP3:channel(dy))composition
role UE(UE,AP,WAAA,IMSI,LUI,AT,LAPID, LNINLNI,
KASME,RES,NUIFunc,UNIDFunc,KDF,PKue,PKap,SND,AP1,RCV,AP1) / role AP(UE,AP,WAAA,SKapws,PKue,PKap,SND,UE2,RCV,UE2,SND_WAAA2,RCV
_WAAA2) / role WAAA(UE,AP,WAAA,LUI,AKAs,IMSI,LNINLNI,XRES,KASME,NUIFunc,UNIDFunc,KDF,SKapws,PKue,PKap,SND,AP3,RCV,AP3) end role
role
environment()def=const
ue:agent,ap:agent,aaaa:agent,imsi,lui,at,lapid,
lninlni,kasme,res,
rue,nui,mws,rnap,unid,nkasme,nres,nmk,akas,xres,nxres:text,nuifunc,unidfunc,kdf:hash_func, pkue,pkap:public_key, skapws:symmetric_key,ue_ap_RNue, ue_ap_NUI,
ue_ap_AT, ue_ap_LAPID, ue_ap_IMSI, ue_ap_NRES, ap_waaa_AT,ap_waaa_RNue, ap_waaa_IMSI,ap_waaa_LAPID, ap_waaa_NUI,ap_ue_RNws,
ap_ue_UNID,ap_waaa_NRES,
ap_ue_AKAs,ap_ue_RNap,aaaa_ap_RNws,aaaa_ap_PKue,aaaa_ap_UNID,aaaa_ap_NMK,aaaa_ap_AKAs,ap_ue_msg_auth_NRES,aaaa_ap_msg_auth_NRES,aaaa
_ap_msg_auth_NUI :protocol_id intruder_knowledge = {nuifunc,unidfunc,kdf,pkue,pkap} composition session1(ue,ap,aaaa,imsi,lui,at,lapid,lninlni,kasme,res,
rue,nui,mws,rnap,unid,nkasme, nres,nmk,akas,xres ,nxres,nuifunc,unidfunc,kdf,pkue,pkap,skapws)end role goal authentication_on waaa_ap_msg_auth_NUI
authentication_on waaa_ap_msg_auth_NRES authentication_on ap_ue_msg_auth_NRES secrecy_of ue_ap_RNue secrecy_of ue_ap_NUI secrecy_of ue_ap_AT
secrecy_of ue_ap_LAPID secrecy_of ue_ap_IMSI secrecy_of ue_ap_NRES secrecy_of ap_waaa_AT secrecy_of ap_waaa_RNue secrecy_of ap_waaa_IMSI secrecy_of
ap_waaa_LAPID secrecy_of ap_waaa_NUI secrecy_of ap_ue_RNws secrecy_of ap_ue_UNID secrecy_of ap_waaa_NRES secrecy_of ap_waaa_RNap secrecy_of
ap_ue_AKAs secrecy_of ap_ue_RNap secrecy_of waaa_ap_AKAs secrecy_of waaa_ap_NMK secrecy_of waaa_ap_UNID secrecy_of waaa_ap_PKue secrecy_of
aaaa_ap_RNws end goal environment()

%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.01s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 1 states, Reachable: 0 states, Translation: 0.01 seconds, Computation: 0.00 seconds

%% % PROTOCOL: REPS-AKA4.1

role role UE (UE,AP,WAAA, LAAA: agent,IMSI,UI,AT,LAPID,WLID,RES,NINNI:text,NUIFunc,KDF:hash_func,PKue,PKap :
public_key,SND,AP,RCV_AP:channel(dy))played_by UE def=local State:nat, RNue,NUI,UNINNI,RNIs,NRES,RNap,MK,AKAs:text const
ue_ap_RNue,ue_ap_AT,ue_ap_LAPID,ue_ap_NUI,ue_ap_WLID,ue_ap_IMSI,ap_ue_msg_auth_NUI,ue_ap_NRES, ap_ue_msg_auth_NRES
,aaaa_ap_msg_auth_NRES:protocol_id init State:= 0 transition 1.State=0 / RCV_AP(start) => State:=1 / RNue' := new() / NUI' := NUIFunc(RNue'.UI) /
SND_AP({RNue'.AT.LAPID,WLID, {NUI'}_inv(PKue),IMSI}_PKap) / secret(RNue'.ue_ap_RNue, {UE, AP}) / secret(AT, ue_ap_AT, {UE, AP}) / secret(LAPID
,ue_ap_LAPID, {UE, AP}) / secret(NUI', ue_ap_NUI, {UE, AP}) / secret(WLID, ue_ap_WLID, {UE, AP}) / secret(IMSI, ue_ap_IMSI, {UE, AP}) / witness (UE,AP,
ap_ue_msg_auth_NUI,NUI') 7. State=1 / RCV_AP({UNINNI'.RNIs'}.PKue) => State:=2 / UNINNI' := NUIFunc (RNIs'.NINNI) / NRES' := KDF(RES.RNIs'.RNue) /
SND_AP({NRES'}.inv(PKue))_PKap) / secret(NRES',ue_ap_NRES, {UE, AP}) / witness (UE,AP, ap_ue_msg_auth_NRES,NRES) / witness (UE,WAAA,
aaaa_ap_msg_auth_NRES,NRES) 11. State=2 / RCV_AP({AKAs'.PKue) => State:=3 / MK' := KDF(UNINNI.UI.NRES) end role role AP(UE,AP,WAAA,
LAAA: agent,SKapws:symmetric_key, PKue,PKap : public_key, SND,UE,RCV,UE,SND_WAAA,RCV_WAAA:channel(dy)) played_by AP def=local State:nat,
RNue,AT,LAPID,WLID,NUI,IMSI,NXRES,UNINNI,RNIs,NRES,MK,AKAs:text const
ap_waaa_AT,ap_waaa_NUI,ap_waaa_LAPID,ap_waaa_IMSI,ap_waaa_WLID,ap_waaa_RNue,ap_ue_msg_auth_NUI,ap_ue_RNIs,ap_ue_UNINNI,ap_waaa_NRES,ap
_ue_msg_auth_NRES, ap_ue_AKAs:protocol_id init State:=0 transition 2.State=0 / RCV_UE({RNue'.AT.LAPID,WLID,{NUI'}_inv(PKue),IMSI}_PKap) => State:=1 /
SND_WAAA({AT,NUI'.IMSI,LAPID,WLID,RNue'}. _SKapws)
/secret(AT,ap_waaa_AT,{AP,WAAA})/secret(NUI',ap_waaa_NUI,{AP,WAAA})/secret(LAPID,ap_waaa_LAPID,{AP,WAAA})/secret(IMSI,ap_waaa_IMSI,{AP,W
AAA})/secret(WLID,ap_waaa_WLID, {AP, WAAA}) / secret(RNue',ap_waaa_RNue, {AP, WAAA}) / request (AP, UE, ap_ue_msg_auth_NUI, NUI') 6. State=1 /
RCV_WAAA({NXRES'.UNINNI'.PKue.RNIs'}. _SKapws) => State:=2 / SND_UE({UNINNI'.RNIs'}. _PKue) / secret(RNIs', ap_ue_RNIs, {AP, UE}) / secret(UNINNI'
, ap_ue_UNINNI, {AP, UE}) 8. State=2 / RCV_UE({NRES'}.inv(PKue))_PKap) => State:=3 / SND_WAAA({NRES'}.inv(PKue))_SKapws) / secret(NRES'
, ap_waaa_NRES, {AP, WAAA}) / request (AP,UE, ap_ue_msg_auth_NRES,NRES) 10. State=3 / RCV_WAAA({AKAs.MK'}. _SKapws) => State:=4 /
SND_UE({AKAs'.PKue) / secret(AKAs, ap_ue_AKAs, {AP, UE}) end role role role WAAA(UE,AP,WAAA, LAAA: agent, AKAs:text, KDF:hash_func,
SKapws,SKwsls:symmetric_key, PKue,PKap,PKIs: public_key, SND,AP,RCV,AP,SND_LAAA,RCV_LAAA:channel(dy)) played by WAAA def=local State:nat,
AT,NUI,IMSI,LAPID,RNue,WLID,NRES,NXRES,UNINNI,RNIs,MK,UI:text const waaa_laiaa_AT, waaa_laiaa_NUI, waaa_laiaa_WLID, waaa_laiaa_IMSI,
aaaa_laiaa_RNue,aaaa_ap_NXRES, waaa_ap_PKue, waaa_ap_UNINNI, waaa_ap_RNIs, waaa_laiaa_msg_auth_UNINNI, waaa_ap_AKAs,
aaaa_ap_MK,aaaa_ap_msg_auth_NRES:protocol_id init State := 0 transition 3.State=0 / RCV_AP({AT,NUI'.IMSI,LAPID,WLID,RNue'}. _SKapws) => State:=1 /
SND_LAAA({AT,NUI'.IMSI,WLID,RNue'}. _SKwsls) / secret(AT,aaaa_laiaa_AT,{WAAA,LAAA}) / secret(NUI',aaaa_laiaa_NUI,{WAAA,LAAA}) / secret(WLID,aaaa
_laiaa_WLID,{WAAA,LAAA}) /
secret(
IMSI,aaaa_laiaa_IMSI,{WAAA,LAAA}) / secret(RNue',aaaa_laiaa_RNue,{WAAA,LAAA}) 5.State=1 / RCV_LAAA({NXRES'}. {UNINNI'}. _inv(PKIs).PKue.RNIs'}. _SK
wsls) => State:=2 / SND_AP
UNINNI'.PKue.RNIs'}. _SKapws) / secret(NXRES',aaaa_ap_NXRES,{WAAA,AP}) / secret(PKue,aaaa_ap_PKue,{WAAA,AP}) / secret(UNINNI',aaaa_ap_UNINNI,{
WAAA,AP}) / secret(RNIs',
aaaa_ap_RNIs,{WAAA,AP}) / request(WAAA,LAAA,aaaa_laiaa_msg_auth_UNINNI,UNINNI') 9.State=2 / RCV_AP({NRES'}.inv(PKue))_SKapws) => State:=3 / M
K' := KDF(UNINNI.UI.NRES) / SND_AP ({AKAs.MK'}. _SKapws) / secret(AKAs, aaaa_ap_AKAs, {WAAA, AP}) / secret(MK', aaaa_ap_MK, {WAAA, AP}) / request
(WAAA,UE, aaaa_ap_msg_auth_NRES,NRES) end role role role LAAA(UE,AP,WAAA, LAAA: agent,
UI,IMSI,NINNI,XRES:text,NUIFunc,KDF:hash_func,SKwsls:symmetric_key,PKue,PKIs : public_key,SND, WAAA,RCV, WAAA:channel(dy))played_by LAAA
def=local State:nat,AT,NUI,WLID,RNue,UNINNI,RNIs,NXRES:text const laaa_waaa_NXRES, laaa_waaa_UNINNI, laaa_waaa_PKue, laaa_waaa_RNIs,
aaaa_laiaa_msg_auth_UNINNI:protocol_id init State := 0 transition 4. State=0 / RCV_WAAA({AT,NUI'.IMSI,WLID,RNue'}. _SKwsls) => State:=1 / RNue' :=
new() / NUI' := NUIFunc(RNue'.UI) / UNINNI' := NUIFunc(RNIs'.NINNI) / NXRES' := KDF(XRES.RNIs'.RNue) /
SND_WAAA({NXRES'}. {UNINNI'}. _inv(PKIs).PKue.RNIs'}. _SKwsls) / secret(NXRES',laaa_waaa_NXRES,{LAAA,WAAA}) / secret(UNINNI',laaa_waaa_UNINNI,
{LAAA, WAAA}) / secret(PKue,laaa_waaa_PKue,{LAAA,WAAA}) / secret(RNIs',laaa_waaa_RNIs,{LAAA,WAAA}) / witness (LAAA, WAAA,
aaaa_laiaa_msg_auth_UNINNI,UNINNI') end role
role session1(UE,AP,WAAA,
LAAA:agent,NXRES,IMSI,UI,AT,LAPID,WLID,RES,XRES,NINNI,RNue,NUI,UNINNI,RNIs,NRES,RNap,MK,AKAs:text,NUIFunc,KDF:hash_func,PKue,PKap,PKI
s:public_key,SKapws,SKwsls:symmetric_key)def=local
SND,AP1,RCV,AP1,SND,UE2,RCV,UE2,SND_WAAA2,RCV_WAAA2,SND,AP3,RCV,AP3,SND_LAAA3,RCV_LAAA3,SND_WAAA4,RCV_WAAA4:channe
l(dy))composition
role UE
(UE,AP,WAAA,LAAA,IMSI,UI,AT,LAPID,WLID,RES,NINNI,NUIFunc,KDF,PKue,PKap,SND,AP1,RCV,AP1) / role AP(UE,AP,WAAA,LAAA,SKapws,PKue,PKa
p,SND,UE2,RCV,UE2,SND_WAAA2,
RCV_WAAA2) / role WAAA(UE,AP,WAAA,LAAA,AKAs,KDF,SKapws,SKwsls,PKue,PKap,PKIs,SND,AP3,RCV,AP3,SND_LAAA3,RCV_LAAA3) / role LAAA(
UE,AP,WAAA,LAAA,UI,IMSI,NINNI,XRES, NUIFunc,KDF,SKwsls,PKue,PKIs,SND_WAAA4,RCV_WAAA4) end role role environment() def=const
ue:agent,ap:agent,aaaa:agent,laiaa:agent,nxres,imsi,ui,at,lapld,wlid,res
uninni,rnls,nres,rnap,mk,akas:text,nuifunc,kdf:hash_func,pkue,pkap,pkls:public_key,skapws,skwsls:symmetric_key,ue_ap_RNue,ue_ap_AT,ue_ap_LAPID,ue_ap_NUI,u
e_ap_WLID,
ue_ap_IMSI,
ap_ue_msg_auth_NUI,ue_ap_NRES,ap_ue_msg_auth_NRES,aaaa_ap_msg_auth_NRES,ap_waaa_AT,ap_waaa_NUI,ap_waaa_LAPID,ap_waaa_IMSI,ap_waaa_WLID

```



```

,ap_waaa_RNue,
ap_ue_RNls,ap_ue_UNINNI,ap_waaa_NRES,ap_ue_AKAs,aaaa_laaa_AT,aaaa_laaa_NUI,aaaa_laaa_WLID,aaaa_laaa_IMSI,aaaa_laaa_RNue,aaaa_ap_NXRES,aaaa
a_ap_PKue,
aaaa_ap_UNINNI,aaaa_ap_RNls,aaaa_laaa_msg_auth_UNINNI,aaaa_ap_AKAs,aaaa_ap_MK,aaaa_waaa_NXRES,laaa_waaa_UNINNI,laaa_waaa_PKue,laaa_waaa_
RNls:protocol_id
intruder_knowledge=(nuifunc,kdf,pkue,pkap,pkls)
composition
session1(ue,ap,aaaa,laaa,nxres,imsi,ui,at,lapid,wlid,res,xres,ninni,rnue,nui,uninni,rnls,nres,rnap,mk,akas,nuifunc,kdf,pkue,pkap,pkls,skapws,skwsls)end role goal
authentication_on ap_ue_msg_auth_NUI authentication_on ap_ue_msg_auth_NRES authentication_on waaa_ue_msg_auth_NRES authentication_on
aaaa_laaa_msg_auth_UNINNI secrecy_of ue_ap_RNue secrecy_of ue_ap_AT secrecy_of ue_ap_LAPID secrecy_of ue_ap_NUI secrecy_of ue_ap_WLID secrecy_of
ue_ap_IMSI secrecy_of ue_ap_NRES secrecy_of ap_waaa_AT secrecy_of ap_waaa_NUI secrecy_of ap_waaa_LAPID secrecy_of ap_waaa_IMSI secrecy_of
ap_waaa_WLID secrecy_of ap_ue_RNls secrecy_of ap_ue_UNINNI secrecy_of ap_waaa_NRES secrecy_of ap_waaa_RNue secrecy_of ap_ue_AKAs secrecy_of
aaaa_laaa_AT secrecy_of waaa_laaa_NUI secrecy_of waaa_laaa_WLID secrecy_of waaa_laaa_IMSI secrecy_of waaa_laaa_RNue secrecy_of waaa_ap_NXRES
secrecy_of waaa_ap_PKue secrecy_of waaa_ap_UNINNI secrecy_of waaa_ap_RNls secrecy_of waaa_ap_AKAs secrecy_of waaa_ap_MK secrecy_of laaa_waaa_NXRES
secrecy_of laaa_waaa_UNINNI secrecy_of laaa_waaa_PKue secrecy_of laaa_waaa_RNls end goal environment()
%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.02s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 9 states, Reachable: 1 states, Translation: 0.03 seconds, Computation: 0.00 seconds
%% % PROTOCOL: REPS-AKA4.2
role role_UE (UE,AP,WAAA, LAAA: agent, IMSI,LUI,AT,LAPID,WLID,LNINLN,RES:text, NUIFunc,UNIDFunc,KDF:hash_func,PKue,PKap :
public_key,SND_AP,RCV_AP:channel(dy))played_by UE def=local State:nat, RNue,RNls,NUI,ULNINLN,NRES,MK,AKAs: text const ue_ap_RNue,ue_ap_LAPID,
ue_ap_AT, ue_ap_NUI, ue_ap_IMSI, ue_ap_WLID, ue_ap_NRES, ap_ue_msg_auth_NRES: protocol_id init State:=0 transition
1.State=0/RCV_AP(start)=>State:=1/RNue:=new()/NUI:=NUIFunc(LUI,RNue)/SND_AP({AT,RNue'.IMSI,NUI'.LAPID,WLID}_PKap)/secret(RNue',
ue_ap_RNue,{UE,AP})/secret(LAPID,ue_ap_LAPID,{UE,AP})/secret(AT,ue_ap_AT,{UE,AP})/secret(NUI,ue_ap_NUI,{UE,AP})/secret(IMSI,ue_ap_IMSI,{UE,A
P})/secret(WLID,ue_ap_WLID,
{UE,AP})9.State=1/RCV_AP({RNls'.ULNINLN'}_PKue)=>State:=2/ULNINLN':=UNIDFunc(LNINLN,RNls')/NRES':=KDF(RES.RNls'.RNue')/SND_AP({{NR
ES'}_inv(PKue)}_PKap)/secret(NRES',ue_ap_NRES,{UE,AP})/ witness (UE,AP, ap_ue_msg_auth_NRES,NRES') 13. State=2 /\ RCV_AP({AKAs}_PKue) =>
State':=3/\ MK':= KDF(ULNINLN,NUI.NRES)end role
LAAA:agent,SKapws:symmetric_key,PKue,PKap:public_key,SND UE,RCV UE,SND_WAAA,RCV_WAAA:channel(dy))played_by AP def=local State:nat,
RNue,AT,IMSI,NUI,LAPID, WLID,RNls, MK,AKAs,ULNINLN,NRES:text const
ap_waaa_AT,ap_waaa_LAPID,ap_waaa_WLID,ap_waaa_NUI,ap_waaa_IMSI,ap_waaa_RNue,ap_ue_RNls,ap_ue_ULNINLN,ap_waaa_NRES,
ap_ue_msg_auth_NRES,ap_ue_AKAs:protocol_id init State:=0 transition
2.State=0/RCV UE({AT,RNue'.IMSI,NUI'.LAPID,WLID}_PKap)=>State:=1/SND_WAAA({AT,RNue'.
IMSI,NUI'.LAPID,WLID}_SKapws)/secret(AT,ap_waaa_AT,{AP,WAAA})/secret(LAPID,ap_waaa_LAPID,{AP,WAAA})/\ secret(WLID,ap_waaa_WLID,{AP,
WAAA})/secret(NUI,ap_waaa_NUI,
{AP,WAAA})/secret(IMSI,ap_waaa_IMSI,{AP,WAAA})/secret(RNue',ap_waaa_RNue,{AP,WAAA})8.State=1/RCV_WAAA({RNls'.PKue.ULNINLN'}_SKapws)
=>State:=2/SND UE({RNls'.
PKue)/secret(RNls',ap_ue_RNls,{AP,UE})/secret(ULNINLN',ap_ue_ULNINLN,{AP,UE})10.State=2/RCV UE({{NRES'}_inv(PKue)}_PKap)=>State':=3/\
SND_WAAA(NRES')_SKapws)/secret(NRES',ap_waaa_NRES,{AP,WAAA}) /\ request (AP,UE, ap_ue_msg_auth_NRES,NRES') 12. State=3 /\
RCV_WAAA({AKAs,MK'}_SKapws) => State':=4 /\SND UE({AKAs}_PKue)/ secret(AKAs,ap_ue_AKAs,{AP,UE}) end role role
role_WAAA(UE,AP,WAAA,LAAA:agent,AKAs:text,KDF:hash_func,SKapws,SKwsls:symmetric_key,PKue,PKap,PKls:public_key, SND_AP,RCV_AP,
SND_LAAA,RCV_LAAA:channel(dy))played_by WAAA def=local State:nat,AT,RNue,IMSI,NUI,LAPID,WLID,RNls,ULNINLN,NXRES,NRES,MK:text const
aaaa_laaa_AT,aaaa_laaa_LAPID,
aaaa_laaa_WLID,aaaa_laaa_NUI,aaaa_laaa_RNue,aaaa_laaa_IMSI,aaaa_ap_RNls,aaaa_ap_PKue,aaaa_ap_ULNINLN,aaaa_laaa_msg_auth_ULNINLN,aaaa_ap
_AKAs,aaaa_ap_MK:
protocol_id State := 0 transition 3. State=0 /\
RCV_AP({AT,RNue'.IMSI,NUI'.LAPID,WLID}_SKapws)=>State:=1/SND_LAAA({AT,RNue'.IMSI,NUI'.LAPID,WLID}_SKwsls)/\ secret(AT,aaaa_laaa_AT,
{WAAA,LAAA})/secret(LAPID,aaaa_laaa_LAPID,{WAAA,LAAA}) /\ secret(WLID,aaaa_laaa_WLID,{WAAA,LAAA}) /\ secret(NUI,aaaa_laaa_NUI,
secret(RNue',aaaa_laaa_RNue,
{WAAA,LAAA})/secret(IMSI,aaaa_laaa_IMSI,{WAAA,LAAA})7.State=1/RCV_LAAA({RNls'.PKue.{ULNINLN'}_inv(PKls).NXRES')_SKwsls)=>State':=2
/NUI:=NUIFunc(LUI,RNue')/ULNINLN':=UNIDFunc(LNINLN,RNls')/NXRES':=KDF(XRES.RNls'.RNue')/SND_WAAA({RNls'.PKue.{ULNINLN'}_inv(PKls),
NXRES')_SKwsls)/secret(RNls', laaa_waaa_RNls,{LAAA,WAAA})/\ secret(PKue,laaa_waaa_PKue,{LAAA,WAAA})/\ secret(ULNINLN',laaa_waaa_ULNINLN,
{LAAA,WAAA})/\ secret(NXRES',laaa_waaa_NXRES,{LAAA,WAAA})/\ witness(LAAA,WAAA,aaaa_laaa_msg_auth_ULNINLN,ULNINLN')end role role
session1(UE,AP,WAAA,LAAA:agent, IMSI,LUI,AT,LAPID,WLID,LNINLN,RES:RNue,RNls,NUI, ULNINLN,NRES,MK,
AKAs,NXRES,XRES:text,NUIFunc,UNIDFunc,KDF:hash_func,PKue,PKap,PKls:public_key,SKapws,SKwsls:symmetric_key)def=local
SND_AP1,RCV_AP1,SND_UE2,RCV_UE2,SND_WAAA2,
RCV_WAAA2,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3,SND_WAAA4,RCV_WAAA4:channel(dy)
composition
role_UE(UE,AP,WAAA,LAAA,IMSI,LUI,AT,LAPID,WLID,LNINLN,RES,NUIFunc,
UNIDFunc,KDF,PKue,PKap,SND_AP1,RCV_AP1)/role_AP(UE,AP,WAAA,LAAA,SKapws,PKue,PKap,SND_UE2,RCV_UE2,SND_WAAA2,RCV_WAAA2)/role_
WAAA(UE,AP,WAAA,LAAA,AKAs,KDF,
SKapws,SKwsls,PKue,PKap,PKls,SND_AP3,RCV_AP3,SND_LAAA3,RCV_LAAA3)/role_LAAA(UE,AP,WAAA,LAAA,LUI,IMSI,LNINLN,XRES,NUIFunc,UNI
DFunc,KDF,SKwsls,PKls,PKue,SND_WAAA4,
RCV_WAAA4) end role role
ue:agent,ap:agent,aaaa:agent,laaa:agent,imsi,lui,at,lapid,wlid,lninln,rs,rnue,rnls,nui,ulninln,nres,mk,akas,nxres,xres:text,
nuifunc,
unidfnc,kdf:hash_func,pkue,pkap,pkls:public_key,skapws,skwsls:symmetric_key,ue_ap_RNue,ue_ap_LAPID,ue_ap_AT,ue_ap_NUI,ue_ap_IMSI,ue_ap_WLID,ue_ap_
NRES,ap_waaa_AT, ap_waaa_LAPID, ap_waaa_WLID,ap_waaa_NUI,ap_waaa_IMSI,ap_waaa_RNue,ap_ue_RNls, ap_ue_ULNINLN,
ap_waaa_NRES,ap_ue_AKAs,aaaa_laaa_AT,
aaaa_laaa_NUI,aaaa_laaa_RNue,aaaa_laaa_IMSI,aaaa_ap_RNls,aaaa_ap_PKue,aaaa_ap_ULNINLN,aaaa_ap_AKAs,aaaa_ap_MK,laaa_waaa_RNls,laaa_waaa_P
Kue,laaa_waaa_ULNINLN,laaa_waaa_NXRES,ap_ue_msg_auth_NRES,aaaa_laaa_msg_auth_ULNINLN:protocol_id
intruder_knowledge=(nuifunc,unidfunc,kdf,pkue,pkap,pkls)composition
session1(ue,ap,aaaa,laaa,
rnls,nui,ulninln,nres,mk,akas,nxres,xres,nuifunc,unidfunc,kdf,pkue,pkap,pkls,skapws,skwsls)end role goal
authentication_on waaa_laaa_msg_auth_ULNINLN secrecy_of ue_ap_RNue secrecy_of ue_ap_LAPID secrecy_of ue_ap_AT secrecy_of ue_ap_NUI secrecy_of
ue_ap_IMSI secrecy_of ue_ap_WLID secrecy_of ue_ap_NRES secrecy_of ap_waaa_AT secrecy_of ap_waaa_LAPID secrecy_of ap_waaa_WLID secrecy_of
ap_waaa_NUI secrecy_of ap_waaa_IMSI secrecy_of ap_waaa_RNue secrecy_of ap_ue_RNls secrecy_of ap_ue_ULNINLN secrecy_of ap_waaa_NRES secrecy_of
ap_ue_AKAs secrecy_of waaa_laaa_AT secrecy_of waaa_laaa_LAPID secrecy_of waaa_laaa_WLID secrecy_of waaa_laaa_NUI secrecy_of waaa_laaa_RNue secrecy_of
aaaa_laaa_IMSI secrecy_of waaa_ap_RNls secrecy_of waaa_ap_PKue secrecy_of waaa_ap_ULNINLN secrecy_of waaa_ap_AKAs secrecy_of waaa_ap_MK secrecy_of
laaa_waaa_RNls secrecy_of laaa_waaa_PKue secrecy_of laaa_waaa_ULNINLN secrecy_of laaa_waaa_NXRES end goal environment()
%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.02s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 4 states, Reachable: 0 states, Translation: 0.02 seconds, Computation: 0.00 seconds
%% % PROTOCOL: REPS-AKA5
role role_IOTGD (IOTGD,IOTGL,HSS,MME: agent, IMSI,FDCID:text,F1, F6, F9, KDF :hash_func,GIK,SKgd:symmetric_key,PKgd,PKgl :
public_key,SND_IOTGL,RCV_IOTGL:channel(dy))played_by IOTGD def=local State:nat,DTID,RNgd,RNgI, RNhss, XRESGD,SKGDL, AUTNGD,AUTNGL: text
const iotgd_iotgl_DTTID, iotgd_iotgl_RNgd, iotgl_iotgd_msg_auth_DTTID,iotgd_iotgl_AUTNGD, iotgd_iotgl_XRESGD,
iotgd_iotgl_msg_auth_AUTNGL,iotgl_iotgd_iotgd_msg_auth_XRESGD: protocol_id init State := 0 transition 1. State=0 /\ RCV_IOTGL(start) => State':=1 /\ RNgd:=new()/
DTID':= F1(IMSI,RNgd'.GIK)/\ SND_IOTGL({{DTID'}_inv(PKgd).RNgd')_PKgl)/\ secret(DTID',iotgd_iotgl_DTTID,{IOTGD,IOTGL}) /\ secret(RNgd',
iotgd_iotgl_RNgd,{IOTGD,IOTGL})/\ witness (IOTGD,IOTGL, iotgl_iotgd_msg_auth_DTTID,DTID')6. State=1 /\
RCV_IOTGL({RNgI'.{AUTNGL'}_inv(PKgl).RNhss')_PKgd) => State':=2/\ XRESGD':= F6(FDCID,RNhss'.SKgd)/\ SKGDL':=KDF(xor(RNgI',RNhss').GIK) /\
AUTNGD':=F9(AUTNGL'.SKGDL') /\ SND_IOTGL({AUTNGD'.XRESGD'}_inv(PKgd))_PKgl) /\ secret(AUTNGD',iotgd_iotgl_AUTNGD,{IOTGD,IOTGL})/\
secret(XRESGD',iotgd_iotgl_XRESGD,{IOTGD,IOTGL})/\ request (IOTGD,IOTGL, iotgd_iotgl_msg_auth_AUTNGL, AUTNGL') /\ witness
(IOTGD,IOTGL,iotgl_iotgd_msg_auth_XRESGD,XRESGD') end role role
agent,IMSI,TM,RDCID,GID,FDCID,RXRESGD,SQN:text,F1,F2,F3,F4,F5,F7,F8,F9,KDF,AUTHFunc:hash_func,GXK,GIK:symmetric_key,PKgd,PKgl,PKmme:public
_key,SND_IOTGD,RCV_IOTGD,
SND_MME,RCV_MME:channel(dy))played_by IOTGL
def=local
State:nat,DTID,RNgd,RNgI,RNhss,SIG,AUTNGD,AUTNGL,MK,CK,IK,AK,KASME,KASMEGL,SKGDL,MXRES,XRESGD,XRESGL, UHNI,HNID: text const
iotgl_mme_RNgI,iotgl_mme_SIG,iotgl_iotgd_msg_auth_DTTID,mme_iotgl_mme_auth_SIG,iotgl_iotgd_AUTNGL,iotgl_mme_msg_auth_AUTNGL,
iotgl_iotgd_msg_auth_AUTNGL,iotgl_mme_XRESGL,iotgl_iotgd_msg_auth_XRESGD,mme_iotgl_mme_auth_XRESGL:protocol_id init State:=0 transition 1.State=0/\
RCV_IOTGD ({DTID'}_inv(PKgd).RNgd')_PKgl) => State':=1 /\ RNgI':=new() /\ SIG':= F1(RNgI'.xor(FDCID, RDCID)) /\
SND_MME({GID'.{SIG'}_inv(PKgd).TM.RNgI')_PKmme) /\ secret(RNgI',
iotgl_mme_RNgI,{IOTGL,

```

```

MME))\secret(SIG',iotgl_mme_SIG,{IOTGL,MME))\request(IOTGL,IOTGD,iotgl_msg_auth_DTID,DTID)\witness(IOTGL,MME,mme_iotgl_msg_auth_SIG,
SIG) 5. State=1/\ RCV_MME( {{AUTNGL'}_inv(PKmm),RNhss'.HNID}_PKgl) => State:=2/\ MK':= KDF(GXK.xor(RNgl, RNhss'))/\ CK':=F3(RNhss'.MK')/\
IK':=F4(RNhss'.MK')/\ AK':=F5(RNhss'.MK') /\ KASME':=KDF(xor(SQN,
AK')\HNID.CK'.IK'.MK')\KASMEGL':=KDF(RNhss'.KASME'.SIG.GXK)\AUTNGL':=F8(SIG.xor(RNgl,RNgd).MK')\SKGDL':=KDF(xor(RNgl,RNhss').GIK)\SND
_IOTGD({RNgl,{AUTNGL'}_inv(PKgl).
}_PKgd)\secret(AUTNGL',iotgl_msg_auth_AUTNGL,{IOTGL,IOTGD))\request(IOTGL,MME,iotgl_mme_msg_auth_AUTNGL,AUTNGL)\witness(IOTGL,IOTGD,iotg
d_iotgl_msg_auth_AUTNGL,
7.State=2/\RCV_IOTGD({AUTNGL',{XRESGD'}_inv(PKgd)}_PKgl)=>State:=3/\AUTNGL':=F9(AUTNGL.SKGDL)\MXRES':=F2(RNhss.MK')\XRESGL':=F7(M
XRESGD')
/\UHNI':=AUTHFunc(AUTNGL.xor(RNhss,RNgl))\SND_MME({{XRESGL'}_inv(PKgl)}_PKmm)\secret(XRESGL',iotgl_mme_XRESGL,{IOTGL,MME))\request
(IOTGL,IOTGD,
iotgl_msg_auth_XRESGD,XRESGD)\witness(IOTGL,MME,mme_iotgl_msg_auth_XRESGL,XRESGL)end role role
role_MME(IOTGD,IOTGL,HSS,MME:agent,HVP,MVP:text,NIFunc,
AUTHFunc:hash_func,SKmh:symmetric_key,PKgl,PKmm,PKhss:public_key,SND_HSS,RCV_HSS,SND_IOTGL,RCV_IOTGL:channel(dy))played_by MME
def=local State:nat, GID,SIG,TM,RNgl, RNhss,XRESGL,AUTNGL,KASMEGL,UHNI,HNID:text const
mme_hss_SIG,mme_hss_RNgl,mme_hss_HNID,mme_iotgl_msg_auth_SIG,hss_mme_msg_auth_HNID, mme_iotgl_AUTNGL,
mme_iotgl_RNhss,mme_iotgl_AUTNGL,iotgl_msg_auth_AUTNGL,mme_iotgl_msg_auth_XRESGL:protocol_id init State:=0
transition
2.State=0
RCV_IOTGL({GID,{SIG'}_inv(PKgl).TM.RNgl'}_PKmm)=>State:=1/\HNID':=NIFunc(xor(HVP,MVP))\SND_HSS({GID.{HNID'}_inv(PKmm),RNgl'.SIG')_SK
mh)\secret(SIG',mme_hss_SIG,
{MME,
HSS})\secret(RNgl',mme_hss_RNgl,{MME,HSS})\secret(HNID',mme_hss_HNID,{MME,HSS})\request(MME,IOTGL,mme_iotgl_msg_auth_SIG,SIG')\witness(M
ME,HSS, hss_mme_msg_auth_HNID, HNID) 4. State=1 /\ RCV_HSS({XRESGL'.{AUTNGL'}_inv(PKhss),KASMEGL'.RNhss'}) => State:=2 /\
SND_IOTGL({{AUTNGL'}_inv(PKmm),RNhss'.HNID}_PKgl)
/\
secret(AUTNGL',mme_iotgl_AUTNGL,{MME,IOTGL})\secret(RNhss',mme_iotgl_RNhss,{MME,IOTGL})\secret(HNID',mme_iotgl_HNID,{MME,IOTGL})\req
ues
t(MME,HSS, mme_hss_msg_auth_AUTNGL,AUTNGL)\witness(MME,IOTGL,iotgl_mme_msg_auth_AUTNGL,AUTNGL')
8.State=2/\RCV_IOTGL({{XRESGL'}_inv(PKgl)}_PKmm)=>State:=3/\UHNI'
:=AUTHFunc
(AUTNGL.xor(RNhss,RNgl))\request(MME,IOTGL,mme_iotgl_msg_auth_XRESGL,XRESGL) end role role role_HSS(IOTGD,IOTGL,HSS,MME: agent, IMSI, HVP,
MVP,RDCID,FDCID,
RXRESGD,SQN:text,KDF,F2,F3,F4,F5,F6,F7,F8,NIFunc:hash_func,SKgd,GXK,SKmh:symmetric_key,PKmm,PKhss:public_key,SND_MME,RCV_MME:channel(dy
)) played_by HSS def=local State :nat,HNID,SIG,RNgl,GID,RNhss,MK,MXRES,XRESGD,XRESGL,KASME,KASMEGL,CK,IK,AK,AUTNGL:text const
hss_mme_XRESGL,hss_mme_AUTNGL,hss_mme_KASMEGL, hss_mme_msg_auth_HNID,mme_hss_msg_auth_AUTNGL:protocol_id init State:= 0 transition
3.State=0/\RCV_MME({GID,{HNID'}_inv(PKmm),RNgl'.SIG')_SKmh)=>State:=1/\ RNhss':=new()\/\ HNID':= NIFunc(xor(HVP,MVP))\/\ MK':=
KDF(GXK.xor(RNgl', RNhss'))\/\ MXRES':=F2(RNhss'.MK')\/\ CK':=F3(RNhss'.MK')\/\ IK':=F4(RNhss'.MK')\/\ AK':=F5(RNhss'.MK')\/\ XRESGD':=
F6(FDCID.RNhss'.SKgd)\XRESGL':=F7(MXRES'.xor(XRESGD',RXRESGD))\KASME':=KDF(xor(SQN,AK').HNID'.CK'.IK'.MK')\KASMEGL':=KDF(RNhss'.KAS
ME'.SIG.GXK)\/\
AUTNGL':=F8(SIG'.xor(RNgl',
RNhss').MK')\SND_MME({{XRESGL'.{AUTNGL'}_inv(PKhss),KASMEGL'.RNhss'}_SKmh)\secret(XRESGL',hss_mme_XRESGL,{HSS,MME})\secret(AUTNGL',
hss_mme_AUTNGL,{HSS,MME})\ secret (KASMEGL',hss_mme_KASMEGL, {HSS, MME})\ request (HSS, MME, hss_mme_msg_auth_HNID, HNID')\ witness
(HSS, MME, mme_hss_msg_auth_AUTNGL, AUTNGL') end role role
session1(IOTGD,IOTGL,HSS,MME:agent,IMSI,FDCID,DTID,RNgd,RNgl,RNhss,XRESGD,SKGDL,AUTNGD,AUTNGL,RDCID,GID,TM,RXRESGD,SQN,SIG,MK
,CK,IK,AK,KASME,KASMEGL,MXRES,XRESGL,UHNI,HNID,HVP,MVP:text,F1,F2,F3,F4,F5,F6,F7,F8,F9,KDF,AUTHFunc,NIFunc:hash_func,PKgl,PKgd,PKhss,
PKmm:public_key,SKgd,GXK,SKmh,GIK:symmetric_key)def=local SND_IOTGL1,
RCV_IOTGL1,SND_IOTGD2,RCV_IOTGD2,SND_MME2,RCV_MME2,SND_IOTGL3,RCV_IOTGL3,SND_HSS3,RCV_HSS3,SND_MME4,RCV_MME4:channel(
dy)
composition
role_IOTGD(IOTGD,IOTGL
,HSS,MME,IMSI,FDCID,F1,F6,F9,KDF,GIK,SKgd,PKgd,PKgl,SND_IOTGL1,RCV_IOTGL1)/\role_IOTGL(IOTGD,IOTGL,HSS,MME,IMSI,RDCID,GID,TM,FDCI
D,RXRESGD,SQN,F1,F2,F3,F4,F5,F7,F8,
F9,KDF,
AUTHFunc,GXK,GIK,PKgd,PKgl,PKmm,SND_IOTGD2,RCV_IOTGD2,SND_MME2,RCV_MME2)/\role_MME(IOTGD,IOTGL,HSS,MME,HVP,MVP,NIFunc,AU
THFunc,SKmh,PKgl,PKmm,PKhss, SND_IOTGL3,RCV_IOTGL3,SND_HSS3,RCV_HSS3) /\role_HSS(IOTGD,IOTGL,HSS,MME, IMSI, HVP,
MVP,RDCID,FDCID, RXRESGD,SQN,KDF, F2, F3, F4, F5,F6, F7, F8, NIFunc,SKgd,GXK,SKmh, PKmm, PKhss,SND_MME4,RCV_MME4)end role role
environment() def=const iotgd:agent,iotgl:agent,mme:agent,hss:agent, imsi,fdcid,dtid,rngd,rngl,rnhss,xresgd, skgd,autngd,autngl, rdcid,gid,
tm,rxresgd,sqn,sig,mk,ck,ik,ak,kasme,kasmevl,mxres,xresgl,uhni,hnid,hvp,mvp:text,f1,f2,f3,f4,f5,f6,f7,f8,f9,kdf,authfunc,nifunc:hash_func,pgkl,pgkd,pkhss,pkmm:publ
ic_key,skgd,gxk,skmh ,gik:symmetric_key,iotgd_iotgl_DTID, iotgd_iotgl_RNgd, iotgl_iotgd_msg_auth_DTID,iotgd_iotgl_AUTNGD,iotgd_iotgl_XRESGD,
iotgd_iotgl_msg_auth_AUTNGL,iotgl_iotgd_msg_auth_XRESGD, iotgl_mme_RNgl, iotgl_mme_SIG, mme_iotgl_msg_auth_SIG,
iotgl_iotgd_AUTNGL,iotgl_mme_msg_auth_AUTNGL,iotgl_mme_XRESGL,mme_iotgl_msg_auth_XRESGL,mme_hss_SIG, mme_hss_RNgl,mme_hss_HNID,
hss_mme_msg_auth_HNID, mme_iotgl_AUTNGL, mme_iotgl_RNhss,mme_iotgl_HNID,mme_hss_msg_auth_AUTNGL,hss_mme_XRESGL, hss_mme_AUTNGL,
hss_mme_KASMEGL :protocol_id intruder_knowledge ={f1,f2,f3,f4,f5,f6,f7,f8,f9,kdf,authfunc,nifunc,pgkl,pgkd,pkhss,pkmm} composition
session1(iotgd,iotgl,mme,hss,imsi,fdcid,dtid, rngd,rngl, rnhss,xresgd,skgd,autngd,autngl,rdcid ,gid,tm,rxresgd,sqn,sig,mk,ck,ik,ak, kasme, kasmevl,
mxres,xresgl,uhni,hnid,hvp,mvp,f1,f2,f3,f4,f5,f6,f7,f8,f9,kdf,authfunc,nifunc,pgkl,pgkd,pkhss,pkmm,skgd,gxk,skmh,gik) end role goal authentication_on
iotgl_iotgd_msg_auth_DTID authentication_on iotgd_iotgl_msg_auth_AUTNGL authentication_on iotgl_iotgd_msg_auth_XRESGD authentication_on
mme_iotgl_msg_auth_SIG authentication_on iotgl_mme_msg_auth_AUTNGLauthentication_on mme_iotgl_msg_auth_XRESGL authentication_on
hss_mme_msg_auth_HNID authentication_on mme_hss_msg_auth_AUTNGL secrecy_of iotgd_iotgl_DTID secrecy_of iotgd_iotgl_RNgd secrecy_of
iotgd_iotgl_AUTNGD secrecy_of iotgd_iotgl_XRESGD secrecy_of iotgl_mme_RNgl secrecy_of iotgl_mme_SIG secrecy_of iotgl_iotgd_AUTNGL secrecy_of
iotgl_mme_XRESGL secrecy_of mme_hss_SIG secrecy_of mme_hss_RNgl secrecy_of mme_hss_HNID secrecy_of mme_iotgl_AUTNGL secrecy_of mme_iotgl_RNhss
secrecy_of mme_iotgl_HNID secrecy_of hss_mme_XRESGL secrecy_of hss_mme_AUTNGL secrecy_of hss_mme_KASMEGL end goal environment()

```

%%Statistics (OFMC mode): parseTime: 0.00s, searchTime: 0.06s, visitedNodes: 2 nodes, depth: 1 piles
%%Statistics (ATSE mode): Analysed: 7 states, Reachable: 1 states, Translation: 0.02 seconds, Computation: 0.00 seconds

Appendix 2: Scyther codes

//Description of REPS-AKA1 Protocol

```
const pk:Function;secret sk:Function;inversekeys (pk,sk);usertype Key;hashfunction H1,H2,H3,H4,H5,H6,H7,H8,H9,H10,H11,H12,AUTHFunc,
KDF,F1,F2,F3,F4,F5,XOR; protocol REPSAKA1(UE,MME,HSS){role UE{var RAND:Nonce;fresh Rk,Ru:Nonce;var kum:Key;const
IMSI,MVP,HVP, SQN,AMF,OSK,SNID:Function;macro NNK=XOR(Rk, OSK);macro HMSI=H1(IMSI);macro HRu=H2(Ru);macro
MVP=H3(MVP);macro UIH=H4 (XOR(IMSI,OSK));macro kum=H5(XOR(IMSI,NNK));macro HSNID=H6(SNID); macro
RNK=H10(UIH);macro MA=H11(RNK,HSNID);macro HNID=H9 (MVP,HVP);macro MAC=F1(SQN,RAND,AMF,RNK);macro
RES=F2(RAND,RNK);macro CK=F3(RAND ,RNK);macro IK=F4(RAND,RNK);macro AK=F5 (RAND,RNK);macro HRES=H12(RES);macro
KASME=KDF(XOR(SQN,AK),HSNID,CK,IK);macro AUTN=AUTHFunc(XOR (SQN, AK),AMF,MAC);macro HRES=H12(RES);macro
A1={HMSI,HRu,HMVP,kum,{NNK}sk(UE)}pk(MME);macro A2={HMSI,HMVP,UIH}pk(HSS);macro A6={SQN,RAND, AUTN,
HMVP,HSNID}kum;macro A7={RNK,MA,AMF}pk(UE);macro A8={HRES}kum;send_uemme1(UE,MME,A1,A2);recv_mmeue1(MME,UE,
A6, A7);send_uemme2 (UE, MME,A8);claim_ue1(UE,Secret,HMSI);
claim_ue2(UE,Secret,HRu);claim_ue3(UE,Secret,HMVP);claim_ue4(UE,Secret,kum);claim_ue5(UE,Secret,NNK);claim_ue6(UE,Secret,UIH);cl
aim_ue7(UE,Secret,HRES);claim_uea(UE,Alive);claim_uew(UE,Weakagree); claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role
MME{var RAND,Rk, Ru:Nonce;var Kmh:Key;const IMSI,MVP,HVP,SQN, kum,AMF,OSK, SNID:Function;macro HSNID=H6(SNID);macro
kmh=H7(XOR(SNID,HMSI));macro HHVP=H8(HVP); macro HNID=H9(MVP,HVP);macro RNK=H10 (UIH);macro
MA=H11(RNK,HSNID);macro HNID=H9(MVP,HVP);macro MAC=F1(SQN,RAND,AMF,RNK);macro RES=F2( RAND,RNK);macro CK=F3
(RAND, RNK);macro IK=F4(RAND,RNK);macro AK=F5(RAND,RNK);macro HRES=H12(RES);macro KASME=KDF(XOR(SQN,AK),
HSNID, CK,IK);macro AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro
A1={HMSI,HRu,HMVP,kum,{NNK}sk(UE)}pk(MME);macro A2={HMSI,HMVP,UIH} pk(HSS); macro A3=
{kmh,HHVP,HSNID,{NNK}sk(UE)}pk(HSS); macro A4={HNID,SQN,RAND,AUTN,KASME,HRES,HMVP}kmh;macro
A5={RNK,MA,AMF} pk(UE);macro A6={SQN,RAND, AUTN,HMVP,HSNID}kum;macro A7={RNK,MA,AMF}pk(UE);macro
A8={HRES}kum;macro HRES=H12(RES); recv_uemme1(UE, MME,A1,A2); send_mmehss1 (MME,
HSS,A2,A3);recv_hssmme1(HSS,MME,{A4,A5});send_mmeue1(MME,UE,A6,A7);recv_uemme2(UE,MME,A8);claim_mme1(MME,Secret,HI
MSI);claim_mme2(MME,Secret,HMVP);claim_mme3(MME,Secret,UIH);claim_mme4(MME,Secret,kmh);claim_mme5(MME,Secret,HHVP);cl
aim_mme6(MME,Secret,HSNID);claim_mme7(MME,Secret,NNK);claim_mme8(MME,Secret,SQN);claim_mme9(MME,Secret,RAND);claim_
mme10(MME,Secret,AUTN);claim_mme11(MME,Secret,HMVP);claim_mme12(MME,Secret,RNK);claim_hss3(HSS,Secret,MA);claim_m
me14(MME,Secret,AMF);claim_mmea(MME,Alive);claim_mmew(MME,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nis
ynch);}role HSS {fresh RAND:Nonce;var Rk,Ru:Nonce;var kmh:Key;const IMSI,MVP,HVP, SQN,AMF,OSK,SNID: Function;macro
NNK=XOR(Rk,OSK);macro HMSI=H1(IMSI);macro HMVP=H3(MVP);macro HHVP=H8(HVP);macro UIH=H4(XOR(IMSI,OSK));macro
RNK=H10(UIH);macro MA=H11(RNK, HSNID);macro HNID=H9(MVP,HVP);macro MAC=F1(SQN,RAND,AMF,RNK); macro
RES=F2(RAND,RNK);macro CK=F3(RAND,RNK);macro IK=F4(RAND,RNK);macro AK= F5(RAND,RNK);macro HRES=H12(RES);macro
KASME=KDF(XOR(SQN,AK),HSNID,CK,IK); macro AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro A2={HMSI, HMVP,
UIH}pk(HSS); macro A3={kmh,HHVP,HSNID,{NNK}sk(UE)}pk(HSS); macro
A4={HNID,SQN,RAND,AUTN,KASME,HRES,HMVP}kmh;macro A5={RNK,MA,AMF}pk (UE); recv_mmehss1(MME,HSS,A2,A3);
send_hssmme1(HSS,MME,{A4,A5});claim_hss1(HSS,Secret,HNID); claim_hss2(HSS,Secret,SQN);claim_hss3(HSS,Secret,RAND);
claim_hss4(HSS,Secret,AUTN);claim_hss5(HSS,Secret,KASME);claim_hss6(HSS,Secret,HRES);claim_hss7(HSS,Secret,HMVP);claim_hss8(H
SS,Secret,RNK);claim_hss9(HSS,Secret,MA);claim_hss10(HSS,Secret,AMF);claim_hssa(HSS,Alive);claim_hssw(HSS,Weakagree);claim_hssni
a(HSS,Niagree);claim_hssnis(HSS,Nisynch);}}
```

//Description of REPS-AKA2 Protocol for X2-handover

```
const pk:Function;secret sk:Function;inversekeys (pk,sk);usertype Key;hashfunction H1,AuthFunc,NKGFunc,KDF,FSKFunc,UKGFunc,XOR;
protocol REPSAKA2(UE, ENB1,ENB2,MME){role UE{var RNe1,RNe2,T4,RNe2P,T2P:Nonce;fresh RNue,T1,RNueP,T1P:Nonce;var
FSKe2u:Key;const UECap,IMSI,eID1,eID2,ASIAD, ASEAD,
ASUPEAD,NASCAD,NASIAD,UECapP,e1ueAuth,Xe1ueAuthP,KeNB,HMSI,e2ueAuth:Function;macro
Xe1ueAuthP=KDF(XOR(e1ueAuth,RNe1));macro HMSI=H1(IMSI);macro e1ueAuthP=AuthFunc(XOR(e1ueAuth,RNue));macro
NKG=NKGFunc (RNe2,RNue,RNe1);macro KeNBP=KDF(KeNB,NKG);macro KRRCint=KDF(KeNBP, XOR(RNue,RNe2),ASIAD);macro
KRRCenc=KDF(KeNBP,XOR(RNue, RNe2),ASEAD);macro KRRCPenc=KDF(KeNBP,XOR(RNue,RNe2),ASUPEAD);macro
KASMEP=KDF( KASME,NKG);macro KNASenc=KDF (KASMEP, NASCAD);macro KNASint=KDF(KASMEP,NASIAD);macro
e2ueAuth=AuthFunc(XOR(RNe1,RNe2));macro e2ueAuthP= AuthFunc(XOR(e2ueAuth, RNueP));macro
FSKe2u=FSKFunc(RNe2,HMSI);macro Xe2ueAuth=AuthFunc(XOR(e2ueAuth,RNe2P));macro UK=UKGFunc(RNe2P,RNueP,
e2ueAuth);macro e2ueAuthPP=AuthFunc(XOR(RNueP,RNe2P));macro KRRCintP=KDF(ASIAD,KRRCint,UK);macro KRRCencP=KDF
(ASEAD, KRRCenc,UK);macro KRRCPencP=KDF(ASUPEAD,KRRCPenc,UK);macro KNASencP=KDF(NASCAD,KNASenc,UK);macro
KNASintP=KDF(NASIAD, KNASint,UK);macro M1={HMSI,UECap, RNue,eID1,pk(UE),{e1ueAuthP}sk(UE)}pk(ENB2);macro
M5={RNe2,RNe1,RNue,T4,{FSKe2u}sk(ENB2),{Xe1ueAuthP}sk(ENB1)}pk(UE);macro M6={RNe2,{e2ueAuth} sk(UE)}pk(ENB2);macro
M1P={HMSI,RNueP,UECapP,eID1,T1P,{e2ueAuthP}sk(UE)}FSKe2u;macro
M2P={RNe2P,RNueP,T2P,{Xe2ueAuth}sk(ENB2)}FSKe2u;send_
ueenb2m1(UE,ENB2,M1);recv_eneb2uem5(ENB2,UE,M5);send_ueenb2m6(UE,ENB2,M6);send_ueenb2m1P(UE,ENB2,M1P);recv_eneb2uem2P
(ENB2,UE,M2P);claim_ue1(UE,Secret,HMSI);claim_ue2(UE,Secret,UECap);claim_ue3(UE,Secret,RNue);claim_ue4(UE,Secret,eID1);claim_u
e5(UE,Secret,e1ueAuthP);claim_ue6(UE,Secret,RNe2);claim_ue7(UE,Secret,e2ueAuth);claim_ue8(UE,Secret,RNueP);claim_ue9(UE,Secret,UE
CapP);claim_ue10(UE,Secret,T1P);claim_ue11(UE,Secret,e2ueAuth);claim_uea(UE,Alive);claim_uew(UE,Weakagree);claim_uenia(UE,Niagr
ee);claim_uenis(UE,Nisynch);}role ENB1{fresh RNe1,T3:Nonce;var RNue,RNe2,T2,RNe2P, T2P: Nonce;const
UECap,eID2,eID1,MVP,e1ueAuth,IMSI,KeNB ,e1ueAuth,KeNB,HMSI:Function;macro HMSI=H1(IMSI); macro e1ueAuthP=AuthFunc
(XOR(e1ueAuth, RNue));macro NKG=NKGFunc(RNe2,RNue, RNe1);macro KeNBP=KDF(KeNB,NKG);macro
Xe1ueAuthP=KDF(XOR(e1ueAuth,RNe1));macro M2={ UECap,RNe2,T2, RNue,eID2,
{HMSI}sk(ENB2),{e1ueAuthP}sk(UE)}pk(ENB1);macro
{HMSI,KeNBP,RNe2,RNe1,RNue,T3,{Xe1ueAuthP}sk(ENB1)}pk(ENB2);macro M4={HMSI,
UECap,eID2,NKG,T3,RNe1,{MVP}sk(ENB1)}pk(MME);recv_eneb2enb1m2(ENB2,ENB1,M2);send_eneb1enb2m3(ENB1,ENB2,M3);send_eneb
1memm4(ENB1,MME,M4);claim_eneb1c1(ENB1,Secret,HMSI);claim_eneb1c2(ENB1,Secret,KeNBP);claim_eneb1c3(ENB1,Secret,RNe2);claim_
eneb1c4(ENB1,Secret,RNe1);claim_eneb1c5(ENB1,Secret,RNue);claim_eneb1c6(ENB1,Secret,T3);claim_eneb1c7(ENB1,Secret,Xe1ueAuthP);clai
m_eneb1c8(ENB1,Secret,UECap);claim_eneb1c9(ENB1,Secret,eID2);claim_eneb1c10(ENB1,Secret,NKG);claim_eneb1c11(ENB1,Secret,MVP);clai
m_eneb1a(ENB1,Alive);claim_eneb1w(ENB1,Weakagree);claim_eneb1nia(ENB1,Niagree);claim_eneb1nis(ENB1,Nisynch);}role ENB2{fresh
RNe2,T2,T4,T2P,RNe2P:Nonce;var RNue,RNe1,T1,T3,RNueP,T1P:Nonce;var FSKe2u:Key;const UECap,eID1,eID2,IMSI,ASIAD,ASEAD,
ASUPEAD,MVP,UECapP,e1ueAuth,KeNB,HMSI:Function;macro HMSI=H1(IMSI);macro
e1ueAuthP=AuthFunc(XOR(e1ueAuth,RNue));macro NKG=NKGFunc (RNe2, RNue,RNe1);macro KeNBP=KDF(KeNB,NKG);macro
Xe1ueAuthP=KDF (XOR(e1ueAuth,RNe1));macro e2ueAuth=AuthFunc(XOR(RNe1,RNe2));macro KRRCint=KDF
(KeNBP,XOR(RNue,RNe2),ASIAD);macro KRRCenc =KDF(KeNBP,XOR(RNue,RNe2),ASEAD);macro
KRRCPenc=KDF(KeNBP,XOR(RNue,RNe2),ASUPEAD);macro FSKe2u= FSKFunc(RNe2,HMSI);macro
e2ueAuth=AuthFunc(XOR(RNe1,RNe2));macro e2ueAuthP=AuthFunc(XOR(e2ueAuth, RNueP));macro UK=UKGFunc(RNe2P,RNueP,
e2ueAuth);macro e2ueAuthPP=AuthFunc(XOR(RNueP,RNe2P));macro KRRCintP=KDF(ASIAD,KRRCint,UK);macro
KRRCPencP=KDF(ASEAD,KRRCenc,UK);macro KRRCPencP=KDF(ASUPEAD,KRRCPenc,UK);macro
```

```

Xe2ueAuth=AuthFunc(XOR(e2ueAuth,RNe2P));macro M1={HIMSI,UECap,RNue,eID1,pk(UE),{e1ueAuthP}sk (UE)}pk(ENB2);macro
M2={UECap,RNe2,T2,RNue,eID2,{HIMSI}sk(ENB2),{e1ueAuthP}sk(UE)}pk(ENB1);macro M3={ HIMSI,KeNPB,RNe2,RNe1, RNue,T3,
{Xe1ueAuthP }sk(ENB1)}pk(ENB2);macro M5={RNe2,RNe1,RNue,T4,{FSKe2u}sk(ENB2),{Xe1ueAuthP }sk(ENB1)}pk(UE);macro
M6={RNe2,{e2ueAuth}sk(UE)}pk(ENB2);macro M1P={ HIMSI,RNueP,UECapP,eID1,T1P,{e2ueAuthP}sk(UE)}FSKe2u;macro M2P
={RNe2P,RNueP,T2P,{Xe2ueAuth}sk(ENB2)}FSKe2u;macro M3P={UECapP,RNe2P,T2P,UK,
HIMSI,{MVP}sk(ENB2)}pk(MME);recv_ueenb2m1(UE,ENB2,M1);send_eneb2m1m2(ENB2,ENB1,M2);recv_eneb2m3(ENB1,ENB2,M3)
;send_eneb2uem5(ENB2,UE,M5);recv_eneb2m6(UE,ENB2,M6);recv_eneb2m1P(UE,ENB2,M1P);send_eneb2uem2P(ENB2,UE,M2P);send_en
b2mmem3P(ENB2,MME,M3P);claim_eneb2c1(ENB2,Secret,UECap);claim_eneb2c2(ENB2,Secret,RNe2);claim_eneb2c3(ENB2,Secret,T2);claim_
eneb2c4(ENB2,Secret,RNue);claim_eneb2c5(ENB2,Secret,eID2);claim_eneb2c6(ENB2,Secret,HIMSI);claim_eneb2c7(ENB2,Secret,e1ueAuthP);cla
im_eneb2c8(ENB2,Secret,RNe2);claim_eneb2c9(ENB2,Secret,RNe1);claim_eneb2c10(ENB2,Secret,RNue);claim_eneb2c11(ENB2,Secret,T4);claim
_eneb2c12(ENB2,Secret,FSKe2u);claim_eneb2c13(ENB2,Secret,Xe1ueAuthP);claim_eneb2c14(ENB2,Secret,RNe2P);claim_eneb2c15(ENB2,Secret
,RNueP);claim_eneb2c16(ENB2,Secret,T2P);claim_eneb2c17(ENB2,Secret,Xe2ueAuth);claim_eneb2c18(ENB2,Secret,UECapP);claim_eneb2c19(E
NB2,Secret,UK);claim_eneb2c20(ENB2,Secret,MVP);claim_eneb2a(ENB2,Live);claim_eneb2w(ENB2,Weakagree);claim_eneb2nia(ENB2,Niagree);
claim_eneb2nis(ENB2,Nisynch);}role MME{var RNNe1,T3,RNue,RNe2,RNe2P,T2P, RNueP;Nonce;const MVP,UECap,eID1,eID2,NASCAD,NASIAD,IMSI,HIMSI,UECapP;Function;macro HIMSI=H1(IMSI);macro
NKG=NKGFunc(RNe2,RNue,RNe1);macro KASMEP=KDF(KASME,NKG);macro KNASenc=KDF(KASMEP,NASCAD);macro KNASint=KDF(KASMEP,NASIAD);macro
KNASintP=KDF(NASIAD,KNASint,UK);macro M4={ HIMSI,UECap,eID2,NKG,T3,RNe1,{MVP}sk(ENB1)}pk(MME); macro M3P={UECapP,RNe2P,T2P,UK,HIMSI,{MVP}sk(ENB2)}pk(MME);recv_eneb1mmem4(ENB1,MME,M4);recv_eneb2mmem3P(ENB2,MME,M
3P);claim_mmea(MME,Alive);claim_mmew(MME,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nisynch);}

```

//Description of REPS-AKA3 Protocol

```

const pk:Function;secret sk:Function;inversekeys (pk,sk);usertype Key:hashfunction
HFunc,UIGFunc,NIFunc,HIGFunc,NKGFunc,F1,F2,F3,F4,F5, XOR,KDF,AUTHFunc, NKENBG,LAFunc,UKFunc,UFunc;protocol
REPSAKA3(UE,MME,HSS,ENB){role UE{var RNmme,RNhs,RAND,RNenb;Nonce;fresh RNue,TS;Nonce;const IMSI,SK,UECap,
HID,MVP,HVP,NASCAD,NASIAD,NASALGID,NASALGC,NASDONE,ASALGC,NASUPC,ASIAD,ASEAD,
ASUPEAD,ASALGID,ASDONE,AMF,SQN,SKmh,SKem:Function;macro HIMSI=HFunc(IMSI,RNue);macro
UI=UIGFunc(HIMSI,SK,RNue);macro HI=HIGFunc(RNhs,UI,SK);macro HNID=NIFunc(XOR(MVP,HVP));macro NKG=NKGFunc(XOR
(RNmme,HI),SK);macro MAC=F1(SQN,RAND,AMF,NKG);macro RES=F2(RAND,NKG);macro CK=F3(RAND,NKG);macro
IK=F4(RAND,NKG);macro AK=F5(RAND, NKG);macro KASME=KDF(XOR(SQN,AK),HNID ,CK,IK,NKG);macro
AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro XRES=F2(RAND,NKG);macro KNASenc=KDF
(KASME,NASCAD,NASALGID);macro KNASint=KDF(KASME,NASIAD,NASALGID);macro
KENB=KDF(KASME,NASUPC,UI,HI);macro NKENB=NKENBG(KENB,RNenb);macro KRRCint=KDF(NKENB,ASIAD,ASALGID);macro
KRRCenc=KDF(NKENB,ASEAD,ASALGID);macro KRRCPenc=KDF(NKENB,ASUPEAD,ASALGID);macro LAUTHP =LAFunc
(HIMSI,RES,RNue);macro UK=UKFunc(LAUTHP,XOR(RNmme,KASME));macro URAND=UFunc(RAND,UK);macro
UAUTN=UFunc(AUTN,UK);macro URES=UFunc (XOR(RES,URAND),UK);macro UKASME=UFunc(KASME,UK);macro
M1={HIMSI}sk(UE),RNue,UI,UECap,pk(UE),TS,HID}pk(MME);macro M4={RAND,AUTN,RNhs,
RNmme,HNID,{HI}sk(MME)}pk(UE);macro M5={RNue,RNmme,{RES}sk(UE)}pk(MME);macro
M6={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro M7={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro
M9={RNenb,{ASALGC}sk(ENB)}pk(UE);macro ReM1={HIMSI,{LAUTHP}sk(UE),RNue,TS}pk(MME);macro
ReM2={URAND,UAUTN,RNue,{RNmme}sk(MME)}pk(UE);macro
ReM3={RNmme,{URES}sk(UE)}pk(MME);send_ueem1 (UE,MME,M1);recv_ueem1 (MME,UE,M4);
send_ueem2(UE,MME,M5);recv_ueem2(MME,UE,M6);send_ueem3(UE,MME,M7);recv_ueem1(ENB,UE,M9);send_ueenb1(UE,ENB,M
10);send_ueem4(UE,MME,ReM1);recv_ueem4(MME,UE,ReM2);send_ueem5(UE,MME,ReM3);claim_ue1(UE,Secret,HIMSI);claim_ue2
(UE,Secret,UI);claim_ue3(UE,Secret,UECap);claim_ue4(UE,Secret,TS);claim_ue5(UE,Secret,HID);claim_ue6(UE,Secret,RNue);claim_ue7(UE,
Secret,RNmme);claim_ue8(UE,Secret,RES);claim_ue9(UE,Secret,NASDONE);claim_ue10(UE,Secret,ASDONE);claim_ue11(UE,Secret,LAUT
HP);claim_ue12(UE,Secret,URES);claim_uea(UE,Alive);claim_uew(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role
MME{var RNue,TS,RNhs,RAND:Nonce;fresh RNmme:Nonce;const IMSI,MVP,HVP,SK,UECap,HID,SQN,AMF,NASCAD, NASIAD,
NASALGID,NASALGC,NASDONE,NASUPC,SKmh,SKem:Function;macro HIMSI=HFunc(IMSI,RNue);macro
UI=UIGFunc(HIMSI,SK,RNue);macro HNID=NIFunc (XOR (MVP,HVP));macro HI=HIGFunc(RNhs,UI,SK);macro
NKG=NKGFunc(XOR(RNmme,HI),SK);macro MAC=F1(SQN,RAND,AMF,NKG);macro XRES=F2(RAND,NKG); macro
CK=F3(RAND,NKG);macro IK=F4(RAND,NKG);macro AK=F5(RAND,NKG);macro
KASME=KDF(XOR(SQN,AK),HNID,CK,IK,NKG);macro AUTN=AUTHFunc (XOR (SQN,AK), AMF,MAC);macro
RES=F2(RAND,NKG);macro KNASenc=KDF(KASME,NASCAD,NASALGID);macro
KNASint=KDF(KASME,NASIAD,NASALGID);macro KENB=KDF(KASME, NASUPC,UI,HI);macro
LAUTHP=LAFunc(HIMSI,RES,RNue);macro UK=UKFunc(LAUTHP,XOR(RNmme,KASME));macro URAND=UFunc(RAND,UK);macro
UAUTN=UFunc (AUTN,UK);macro UXRES=UFunc(XOR(XRES,URAND),UK);macro UKASME=UFunc(KASME, UK);macro
URES=UFunc(XOR(RES,URAND),UK);macro M1={HIMSI} sk (UE),RNue,UI,UECap,pk(UE),TS,HID}pk(MME);macro
M2={RNmme,RNue,{HNID}sk(MME),HIMSI,UI}SKmh;macro M3={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro
M4={RAND,AUTN,RNhs,RNmme,HNID,{HI}sk(MME)}pk(UE);macro
M5={RNue,RNmme,{RES}sk(UE)}pk(MME);macro M6={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro
M7={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro ReM1={HIMSI,{LAUTHP}sk(UE),RNue,TS}pk(MME);macro
ReM2={ URAND,UAUTN,RNue,{RNmme}sk(MME)}pk(UE);macro
ReM3={RNmme,{URES}sk(UE)}pk(MME);recv_ueem1(UE,MME,M1);send_mme1(HSS,MME,M3);send_mme2(UE,MME,M5);send_mme3(UE,MME,
M7);send_mme4(UE,MME,ReM1);send_mme5(UE,MME,ReM2);recv_ueem5(UE,MME,ReM3);claim_
mme1(MME,Secret,RNmme);claim_mme2(MME,Secret,RNue);claim_mme3(MME,Secret,HNID);claim_mme4(MME,Secret,HIMSI);claim_m
me5(MME,Secret,UI);claim_mme6(MME,Secret,RAND);claim_mme7(MME,Secret,AUTN);claim_mme8(MME,Secret,RNhs);claim_mme9(M
ME,Secret,NASALGC);claim_mme10(MME,Secret,KENB);claim_mme11(MME,Secret,URAND);claim_mme12(MME,Secret,UAUTN);claim_
mmea(MME,Alive);claim_mmew(MME,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nisynch);}role HSS{var
RNue,RNmme:Nonce;fresh RNhs,RAND:Nonce;const IMSI,MVP,HVP,SK,SQN,AMF,SKmh,SKem:Function;macro
HNID=NIFunc(XOR(MVP,HVP));macro HIMSI= HFunc(IMSI,RNue);macro UI=UIGFunc(HIMSI,SK,RNue);macro
HI=HIGFunc(RNhs,UI,SK);macro NKG=NKGFunc(XOR(RNmme,HI),SK);macro MAC= F1(SQN,RAND, AMF,NKG);macro
XRES=F2(RAND,NKG);macro CK=F3(RAND,NKG);macro IK=F4(RAND,NKG);macro AK=F5(RAND,NKG);macro KASME=KDF(XOR
(SQN,AK),HNID,CK, IK,NKG);macro AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro
M2={RNmme,RNue,{HNID}sk(MME),HIMSI,UI}SKmh;macro M3={HIMSI,LAUTHP}sk(UE),RNue,TS}pk(MME);macro
XRES,RNhs,RNmme}SKmh;recv_mme1(HSS,MME,M2);send_hss1(HSS,MME,M3);claim_hss1(HSS,Secret,HI);claim_hss2(HSS,Sec
ret,RAND);claim_hss3(HSS,Secret,AUTN);claim_hss4(HSS,Secret,KASME);claim_hss5(HSS,Secret,XRES);claim_hss6(HSS,Secret,RNhs);cl
aim_hss7(HSS,Secret,RNmme);claim_hssa(HSS,Alive);claim_hssw(HSS,Weakagree);claim_hssnia
(HSS,Niagree);claim_hssnis(HSS,Nisynch);}role ENB{var RNue,RNmme,RNhs,RAND:Nonce;fresh RNenb:Nonce;const
IMSI,MVP,HVP,SK,UECap,HID,SQN,AMF,NASUPC,ASALGID,ASIAD,ASEAD,ASUPEAD,ASALGC,ASDONE,SKmh,SKem:Function;ma
cro HIMSI=HFunc(IMSI,RNue);macro UI =UIGFunc(HIMSI,SK,RNue);macro HNID=NIFunc(XOR(MVP,HVP));macro
HI=HIGFunc(RNhs,UI,SK);macro NKG=NKGFunc(XOR(RNmme,HI),SK);macro MAC=F1(SQN, RAND,AMF,NKG);macro
CK=F3(RAND,NKG);macro IK=F4(RAND,NKG);macro AK=F5(RAND,NKG);macro KASME=KDF(XOR
(SQN,AK),HNID,CK,IK,NKG);macro AUTN= AUTHFunc(XOR(SQN,AK),AMF,MAC);macro KENB=KDF(KASME,NASUPC,UI,HI);macro
NKENB=NKENBG(KENB,RNenb);macro KRRCint=KDF(NKENB,ASIAD,ASALGID); macro
KRRCenc=KDF(NKENB,ASEAD,ASALGID);macro KRRCPenc=KDF(NKENB,ASUPEAD,ASALGID);macro

```

```

M8={ {KENB}sk(MME)}SKem;macro
M10={ {ASDONE}sk(UE)}pk(ENB);recv_mmeenb1(MME,ENB,M8);send_enbue1(ENB,UE,M9);recv_ueenb1(UE,ENB,M10);
claim_enb1(ENB, Secret,RNenb);claim_enb2(ENB,Secret,ASALGC);claim_enba(ENB,Alive);claim_enbw(ENB,Weakagree);
claim_enbnia(ENB,Niagree);claim_enbnis(ENB,Nisynch);}}
//Description of REPS-AKA4.1.1 Protocol
const pk:Function;secret sk:Function;inversekeys (pk,sk);usertype Key;hashfunction
UIGFunc,NIFunc,NIGFunc,NKGFunc,KDF,F1,F2,F3,F4,F5,XOR, AUTHFunc;protocol REPSAKA411(UE,MME,HSS){role UE{var
RAND,RNmme,RNhss:Nonce;fresh RNue,TS:Nonce;const IMSI,SK,MVP,HVP,SQN,AMF,SKmh:Function;macro UI=UIGFunc(
IMSI,XOR(SK,RNue));macro HNID=NIFunc(XOR(MVP,HVP));macro NI=NIGFunc(RNhss,UI,SK);macro UKG
=NKGFunc(XOR(RNmme,NI));macro MAC=F1(SQN,RAND,AMF,UKG);macro RES=F2(RAND,UKG);macro CK=F3(RAND,UKG);macro
IK=F4(RAND,KG);macro AK=F5(RAND,UKG);macro KASME=KDF(XOR(SQN,AK),HNID,CK,IK, UKG);macro
AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro M1={ {IMSI}sk(UE),RNue,UI,pk(UE),TS}pk(MME);macro
M4={RAND,AUTN,RNhss,RNmme,HNID,{NI}sk (MME)}pk(UE);macro
M5={RNmme,{RES}sk(UE)}pk(MME);send_ueemme1(UE,MME,M1);recv_mmeue1(MME,UE,M4);send_ueemme2(UE,MME,M5);claim_ue1(
UE, Secret,IMSI);claim_ue2(UE,Secret,RNue);claim_ue3(UE,Secret,UI);claim_ue4(UE,Secret,TS);claim_ue5(UE,Secret,RES);claim_uea(UE,Alive);
claim_uew(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role MME{var RAND,RNue,RNhss,TS:Nonce;fresh
RNmme:Nonce;const IMSI,SK,MVP,HVP,SQN ,AMF,SKmh: Function;macro HNID=NIFunc(XOR(MVP,HVP));macro
UI=UIGFunc(IMSI,XOR(SK,RNue));macro NI=NIGFunc(RNhss,UI,SK);macro UKG=NKGFunc(XOR(RNmme,NI)); macro
MAC=F1(SQN,RAND,AMF,UKG);macro XRES=F2(RAND,UKG);macro CK=F3(RAND,UKG);macro IK=F4( RAND,UKG);macro
AK=F5(RAND,UKG);macro KASME =KDF(XOR(SQN,AK),HNID,CK,IK,UKG);macro
AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro RES=F2(RAND,UKG);macro M1={ {IMSI}sk(UE),RNue,UI,pk(UE),TS}pk
(MME);macro M2={RNmme,RNue,{HNID}sk(MME),IMSI,UI}SKmh;macro
M3={ {NI}sk(HSS),RAND,AUTN,KASME,XRES,RNhss,RNmme}SKmh;macro
RNhss,RNmme,HNID,{NI}sk(MME)}pk(UE);macro
M5={RNmme,{RES}sk(UE)}pk(MME);recv_ueemme1(UE,MME,M1);send_mmehss1(MME,HSS,M2);recv_hssmme1
(HSS,MME,M3);send_mmeue1(MME,UE,M4);recv_ueemme2(UE,MME,M5);claim_mme1(MME,Secret,RNmme);claim_mme2(MME,Secret,R
Nue);claim_mme3(MME,Secret,HNID);claim_mme4(MME,Secret,IMSI);claim_mme5(MME,Secret,UI);claim_mme6(MME,Secret,RAND);clai
m_mme7(MME,Secret,AUTN);claim_mme8(MME,Secret,RNhss);claim_mme9(MME,Secret,NI);claim_mmea(MME,Alive);claim_mmw(MM
E,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nisynch);}role HSS{var RNue,RNmme,TS:Nonce;fresh
RNhss,RAND:Nonce;const IMSI,SK,MVP,HVP,SQN,AMF,SKmh:Function;macro HNID=NIFunc(XOR(MVP,HVP));macro
NI=NIGFunc(RNhss,UI,SK);macro UKG=NKGFunc(XOR(RNmme,NI));macro MAC=F1(SQN,RAND,AMF,UKG);macro
XRES=F2(RAND,UKG);macro CK=F3(RAND,UKG); macro IK=F4(RAND,UKG);macro KASME=
KDF(XOR(SQN,AK),HNID,CK,IK,UKG);macro
AUTN=AUTHFunc(XOR(SQN,AK),AMF,MAC);macro
M2={RNmme,RNue,{HNID}sk(MME),IMSI,UI}SKmh;macro
M3={ {NI}sk(HSS),RAND,AUTN,KASME,XRES,RNhss,RNmme}SKmh;recv_mmehss1(MME,HSS,M2); send_
hssmme1(HSS,MME,M3);claim_hss1(HSS,Secret,NI);claim_hss2(HSS,Secret,RAND);claim_hss3(HSS,Secret,AUTN);claim_hss4(HSS,Secret,
KASME);claim_hss5(HSS,Secret,XRES);claim_hss6(HSS,Secret,RNhss);claim_hss7(HSS,Secret,RNmme);claim_hssa(HSS,Alive);claim_hssw(
HSS,Weakagree);claim_hssnia(HSS,Niagree);claim_hssnis(HSS,Nisynch);}}
//Description of REPS-AKA4.1.2 Protocol
const pk:Function;secret sk:Function;inversekeys (pk,sk);usertype Key;hashfunction NUIFunc,UPKFunc,NUIGFunc,NNIGFunc,KDF;protocol
REPSAKA412(UE, MME,HSS){role UE{var RNmme:Nonce;fresh RNue:Nonce;const
IMSI,UI,NI,SK,KASME,RES,AUTN,RAND,SKmh:Function;macro NUI=NUIFunc(UI,RNue);macro NNI= NNIGFunc(NI,NUI,SK);macro
UPK=UPKFunc(NUI,RNmme);macro UKASME=KDF(KASME,UPK);macro NRES= KDF(RES,UPK);macro
UAUTN=KDF(AUTN,UPK);macro URAND=KDF(RAND,UPK,RNmme);macro ReM1={ {NUI}sk(UE),RNue,IMSI}pk(MME);macro
ReM3={RNmme,Nue,{UAUTN}sk(MME),URAND}pk(UE);macro ReM4=
{RNmme,{NRES}sk(UE)}pk(MME);send_ueemme1(UE,MME,ReM1);recv_mmeue1(MME,UE,ReM3);send_ueemme2(UE,MME,ReM4);claim_
ue1(UE,Secret,IMSI);claim_ue2(UE,Secret,RNue);claim_ue3(UE,Secret,NUI);claim_ue4(UE,Secret,NRES);claim_uea(UE,Alive);claim_uew(UE,
Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role MME{var RNue:Nonce;fresh RNmme:Nonce;const
IMSI,SK,UI,AUTN,RAND,KASME,NI,XRES,RES,SKmh:Function;macro NUI=NUIFunc(UI,RNue);macro UPK=
UPKFunc(NUI,RNmme);macro UKASME=KDF(KASME,UPK);macro NRES=KDF(RES,UPK);macro NXRES=KDF(XRES,UPK);macro
UAUTN=KDF(AUTN,UPK);macro URAND=KDF(RAND,UPK,RNmme);macro ReM1={ {NUI}sk(UE),RNue,IMSI}pk(MME);macro
ReM2={RNue,{NUI}sk(MME),IMSI}SKmh; macro ReM3={RNmme,RNue, {UAUTN}sk(MME),URAND}pk(UE);macro
ReM4={RNmme,{NRES}sk(UE)}pk(MME);recv_ueemme1(UE,MME,ReM1);send_mmehss1(MME,HSS, ReM2);send_mmeue1
(MME,UE,ReM3);recv_ueemme2(UE,MME,ReM4);claim_mme1(MME,Secret,RNue);claim_mme2(MME,Secret,NUI);claim_mme3(MME,Secr
et,IMSI);claim_mme4(MME,Secret,RNmme);claim_mme5(MME,Secret,UAUTN);claim_mme6(MME,Secret,URAND);claim_mmea(MME,Ali
ve);claim_mmw(MME,Weakagree);claim_mmenia(MME,Niagree);claim_mmenis(MME,Nisynch);}role HSS{var RNue,RNmme:Nonce;const
IMSI,SK,UI,AUTN,RAND,KASME,NI,XRES,RES,SKmh:Function;macro NUI=
NUIFunc(UI,RNue);macro
NNI=NNIGFunc(NI,NUI,SK);macro ReM2={RNue,{NUI}sk(MME),IMSI}SKmh;recv_mmehss1(MME,HSS,ReM2);claim_hssa(HSS,Alive);
claim_hssw(HSS,Weakagree);claim_hssnia(HSS,Niagree);claim_hssnis(HSS,Nisynch);}}
//Description of REPS-AKA4.2.1 Protocol
const pk:Function;secret sk:Function;inversekeys (pk,sk);hashfunction KDF;protocol REPSAKA421(UE,AP,WAAA,LAAA,HSS){role UE{var
RNls,RNap :Nonce;fresh RNue:Nonce;const HNID,AT,NINNI,IMSI,APID,XRES,UI,RES,AKASuccess:Function;macro
NRES=KDF(RES,RNls);macro MK=KDF(UI,NRES,RNls);macro M1={RNue,HNID ,AT,{NINNI}sk(UE),IMSI}pk(AP);macro
M8={RNls,UI,RNap}pk(UE);macro M9={RNap,{NRES}sk(UE)}pk(AP); macro M12={AKASuccess}pk(UE); send_ueap1(UE,AP,M1);
recv_apue1(AP,UE,M8);send_ueap2(UE,AP,M9);recv_apue2(AP,UE,M12);claim_ue1 (UE,Secret,RNue);claim_ue2(UE,Secret,HNID);
claim_ue3(UE,Secret,AT);
claim_ue4(UE,Secret,NINNI);claim_ue5(UE,Secret,IMSI);claim_ue6(UE,Secret,RNap);claim_ue7(UE,Secret,NRES);claim_uea(UE,Alive);clai
m_uew(UE,Weakagree); claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role AP{var RNls,RNue:Nonce;fresh RNap:Nonce;secret
SKapws:Function;const HNID,AT,NINNI,IMSI,APID,XRES,UI ,RES,AKASuccess:Function;macro NRES=KDF(RES,RNls);macro
M1={RNue,HNID,AT,{NINNI}sk(UE),IMSI}pk(AP);macro M2={AT,HNID,{NINNI}sk(UE), IMSI,APID} SKapws;macro M7={
RNls,pk(UE),UI}SKapws;macro M8={RNls,UI,RNap}pk(UE);macro M9={RNap,{NRES}sk(UE)}pk(AP);macro
M10={ {NRES}sk(UE)}SKapws;macro M11 ={AKASuccess,MK}SKapws;macro
M12={AKASuccess}pk(UE);recv_ueap1(UE,AP,M1);send_apwaaa1(AP,WAAA,M2);recv_waaaap1(WAAA,AP, M7);send_apue1
(AP,UE,M8);recv_ueap2(UE,AP,M9);send_apwaaa2(AP,WAAA,M10);recv_waaaap2(WAAA,AP,M11);send_apue2(AP,UE,M12);claim_ap1(A
P,Secret,HNID);claim_ap2(AP,Secret,AT);claim_ap3(AP,Secret,NINNI);claim_ap4(AP,Secret,RNls);claim_ap5(AP,Secret,IMSI);claim_ap6(AP
,Secret,APID);claim_ap7(AP,Secret,UI);claim_ap8(AP,Secret,RNap);claim_ap9(AP,Secret,NRES);claim_ap10(AP,Secret,AKASuccess);claim_a
pa(AP,Alive);claim_apw(AP,Weakagree);claim_apnia(AP,Niagree);claim_apnis(AP,Nisynch);}role WAAA{var RNls,RNue,RNap:Nonce;secret
SKapws,SKwls:Function;const HNID,AT,NINNI,IMSI,APID,XRES,UI,RES,AKASuccess:Function;macro NXRES =KDF(XRES,RNls);macro
NRES= KDF(RES,RNls);macro MK=KDF(UI,NRES,RNls);macro M2={AT,HNID,{NINNI}sk(UE),IMSI,APID}SKapws;macro
M3={AT,HNID, {NINNI}sk(UE),IMSI} SKwls;macro M6={NXRES,RNls,pk(UE),UI}SKwls;macro M7={RNls,pk(UE),UI}SKapws;macro
M10={ {NRES}sk(UE)}SKapws;macro M11={
AKASuccess,MK}SKapws;recv_apwaaa1(AP,WAAA,M2);send_waaaalaa1(WAAA,LAAA,M3);recv_laawaaa1(LAAA,WAAA,M6);send_waa
aap1(WAAA,AP,M7);recv_apwaaa2(AP,WAAA,M10);send_waaaap2(WAAA,AP,M11);claim_waaa1(WAAA,Secret,AT);claim_waaa2(WAAA
,Secret,NINNI);claim_waaa3(WAAA,Secret,UI);claim_waaa4(WAAA,Secret,IMSI);claim_waaa5(WAAA,Secret,HNID);claim_waaa6(WAAA,
Secret,RNls);claim_waaa7(WAAA,Secret,MK);claim_waaa8(WAAA,Secret,AKASuccess);claim_waaa9(WAAA,Alive);claim_waaaaw(WAAA,

```



```

AKASuccess}pk(UE);recv_ueap1(UE,AP,M1);send_apwaaa1(AP,WAAA,M2);recv_waaaap1(WAAA,AP,M5);send_apue1(AP,UE,M6);recv_ue
ap2(UE,AP,M7);send_apwaaa2(AP,WAAA,M8);recv_waaaap2(WAAA,AP,M9);send_apue2(AP,UE,M10);claim_ap1(AP,Secret,HNID);claim_
ap2(AP,Secret,AT);claim_ap3(AP,Secret,LUI);claim_ap4(AP,Secret,RNue);claim_ap5(AP,Secret,IMSI);claim_ap6(AP,Secret,RNws);claim_ap7
(AP,Secret,RNls);claim_ap8(AP,Secret,NAUTN);claim_ap9(AP,Secret,NLNI);
claim_ap10(AP,Secret,NRES);claim_ap11(AP,Secret,AKASuccess);claim_apa(AP,Alive);claim_apw(AP,Weakagree);claim_apnia(AP,Niagree);
claim_apnls(AP,Nisynch);}role WAAA{var RNue,RNls:Nonce;fresh RNws:Nonce;secret SKapws,SKwsls:Function;const
IMSI,PLUI,AT,HNID,RES,XRES,AUTN,KASME,LNI,AKASuccess:Function;macro LUI=LUIFunc(IMSI,PLUI,RNue);macro
UK=UKFunc(LUI,RNls,RNws);macro NXRES=KDF(XRES,UK);macro NAUTN=KDF(AUTN,UK);macro NKASME=KDF (KASME
,UK);macro NLNI=KDF(LNI,UK);macro NRES=KDF(RES,UK);macro MK=KDF(NKASME,RNws);macro
M2={HNID,AT,LUI,RNue,{IMSI}sk(AP)}SKapws;macro M3={HNID, AT,LUI,RNue,RNws,{IMSI}sk(AP)}SKwsls;macro
M4={RNls,NAUTN,NKASME,NLNI,NXRES}SKwsls;macro M5={RNws,RNls,NAUTN,NLNI}SKapws;macro M8={NRES} SKapws;macro
M9={AKASuccess,MK}SKapws;recv_apwaaa1(AP,WAAA,M2);send_waaa1(WAAA,LAAA,M3);recv_laaa1(LAAA,WAAA,M4);sen
d_waaaap1
(WAAA,AP,M5);recv_apwaaa2(AP,WAAA,M8);send_waaaap2(WAAA,AP,M9);claim_waaa1(WAAA,Secret,HNID);claim_waaa2(WAAA,Sec
ret,AT);claim_waaa3(WAAA,
Secret,LUI);claim_waaa4(WAAA,Secret,RNue);claim_waaa5(WAAA,Secret,RNws);claim_waaa6(WAAA,Secret,IMSI);claim_waaa7(WAAA,S
ecret,RNls);claim_waaa8(WAAA,
Secret,NAUTN);claim_waaa9(WAAA,Secret,NLNI);claim_waaa10(WAAA,Alive);claim_waaa11(WAAA,Weakagree);claim_waaa12(WAAA,Ni
agree);claim_waaa13(WAAA,Nisynch);}role LAAA {var RNue,RNws:Nonce;fresh RNls:Nonce;secret SKwsls:Function;const
IMSI,PLUI,AT,HNID,RES,XRES,AUTN,KASME,LNI,AKASuccess:Function;macro LUI=LUIFunc(IMSI,PLUI,RNue);macro
UK=UKFunc(LUI,RNls,RNws);macro NXRES=KDF(XRES,UK);macro NAUTN=KDF(AUTN,UK);macro NKASME=KDF
(KASME,UK);macro NLNI=KDF(LNI,UK);macro M3={HNID,AT,LUI,RNue,RNws,{IMSI}sk(AP)}SKwsls;macro
M4={RNls,NAUTN,NKASME,NLNI,NXRES}SKwsls;recv_
waalaaa1(WAAA,LAAA,M3);send_laaa1(LAAA,WAAA,M4);claim_laaa1(LAAA,Secret,RNls);claim_laaa2(LAAA,Secret,NAUTN);clai
m_laaa3(LAAA,Secret,NKASME);claim_laaa4(LAAA,Secret,NLNI);claim_laaa5(LAAA,Secret,NXRES);claim_laaa6(LAAA,Alive);claim_laaa
7(LAAA,Weakagree);claim_laaa8(LAAA,Niagree);claim_laaa9(LAAA,Nisynch);}}

```

//Description of REPS-AKA4.3.1 Protocol

```

const pk:Function;secret sk:Function;inversekeys (pk,sk);hashfunction KDF;protocol REPSAKA431(UE,AP2,WAAA){role UE{var
RNws,RNap2:Nonce;fresh RNue:Nonce;const IMSI,AT,LAPID,NINNI,RES,UI,AKASuccess:Function;macro
NRES=KDF(RES,RNue,RNws);macro MK=KDF(NINNI,UI,NRES);macro M1={RNue,AT,LAPID,{NINNI}sk (UE),IMSI}pk(AP2);macro
M4={RNap2,RNws}pk(UE);macro M5={RNue,RNap2,{NRES}sk(UE)}pk(AP2);macro
M8={AKASuccess}pk(UE);send_ueap21(UE,AP2,M1);recv_
ap2ue1(AP2,UE,M4);send_ueap22(UE,AP2,M5);recv_ap2ue2(AP2,UE,M8);claim_ue1(UE,Secret,RNue);claim_ue2(UE,Secret,AT);claim_ue3(
UE,Secret,LAPID);claim_ue4(UE,Secret,NINNI);claim_ue5(UE,Secret,IMSI);claim_ue6(UE,Secret,RNap2);claim_ue7(UE,Secret,NRES);claim_
_uea(UE,Alive);claim_uew(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role AP2{var RNws,RNue:Nonce;fresh
RNap2:Nonce;secret SKap2ws:Function;const IMSI,AT,LAPID,NINNI,RES,UI, AKASuccess:Function;macro
NRES=KDF(RES,RNue,RNws);macro MK=KDF(NINNI,UI,NRES);macro M1={RNue,AT,LAPID,{NINNI}sk(UE),IMSI}pk(AP2);macro
M2={AT,{NINNI}sk(AP2),IMSI,LAPID,RNap2}SKap2ws;macro M3={RNap2,RNws,pk(UE)}SKap2ws;macro
M4={RNap2,RNws}pk(UE);macro M5={RNue,RNap2,{NRES}sk(UE)}pk (AP2);macro M6={RNue,{NRES}sk(UE)}SKap2ws;macro
M7={AKASuccess,MK}SKap2ws;macro M8={AKASuccess}pk(UE);recv_ueap21(UE,AP2,M1);send_ap2waaa1
(AP2,WAAA,M2);recv_waaaap21(WAAA,AP2,M3);send_ap2ue1(AP2,UE,M4);recv_ueap22(UE,AP2,M5);send_ap2waaa2(AP2,WAAA,M6);r
ecv_waaaap22(WAAA,AP2,M7);send_ap2ue2(AP2,UE,M8);claim_ap21(AP2,Secret,RNue);claim_ap22(AP2,Secret,AT);claim_ap23(AP2,Secre
t,LAPID);claim_ap24(AP2,Secret,NINNI);claim_ap25
(AP2,Secret,IMSI);claim_ap26(AP2,Secret,RNap2);claim_ap27(AP2,Secret,RNws);claim_ap28(AP2,Secret,NRES);claim_ap29(AP2,Secret,AK
ASuccess);claim_ap2a(AP2,Alive);claim_ap2w(AP2,Weakagree);claim_ap2nia(AP2,Niagree);claim_ap2nis(AP2,Nisynch);}role WAAA{var
RNap2,RNue:Nonce;fresh RNws:Nonce;secret SKap2ws :Function;const IMSI,AT,LAPID,NINNI,RES,UI,XRES,AKASuccess:Function;macro
NRES=KDF(RES,RNue,RNws);macro NXRES=KDF(XRES,RNue,RNws);macro MK=KDF (NINNI,UI,NRES);macro
M2={AT,{NINNI}sk(AP2),IMSI,LAPID,RNap2}SKap2ws;macro M3={RNap2,RNws,pk(UE)}SKap2ws;macro
M6={RNue,{NRES}sk(UE)}SKap2ws;macro M7={AKASuccess,MK}SKap2ws;recv_ap2waaa1(AP2,WAAA,M2);send_waaaap21(WAAA,AP2,M3);recv_ap2waaa2(AP2,WAAA,M6);
send_waaaap22(WAAA,
AP2,M7);claim_waaa1(WAAA,Secret,RNap2);claim_waaa2(WAAA,Secret,RNws);claim_waaa3(WAAA,Secret,AKASuccess);claim_waaa4(W
AAA,Secret,MK);claim_waaa5(WAAA,Alive);claim_waaa6(WAAA,Weakagree);claim_waaa7(WAAA,Niagree);claim_waaa8(WAAA,Nisyn
ch);}}

```

//Description of REPS-AKA4.3.2 Protocol

```

const pk:Function;secret sk:Function;inversekeys (pk,sk);hashfunction KDF,NUIFunc,UNIDFunc;protocol REPSAKA432(UE,AP2,WAAA){role
UE{var RNws,RNap2:Nonce;fresh RNue:Nonce;const LUI,AT,IMSI,LAPID,LNINLNI,KASME,RES,AKASuccess:Function;macro
NUI=NUIFunc(LUI,RNue);macro UNID=UNIDFunc(LNINLNI,RNws);macro NKASME=KDF(KASME,RNws);macro
NRES=KDF(RES,RNue,RNws);macro NMK=KDF(NKASME,RNap2);macro M1={AT,RNue,IMSI,NUI,LAPID}pk(AP2);macro
M4={RNws,UNID}pk(UE);macro M5={NRES}sk(UE)}pk(AP2);macro
M8={AKASuccess,RNap2}pk(UE);send_ueap21(UE,AP2,M1);recv_ap2ue1(AP2,UE,M4);send_
ueap22(UE,AP2,M5);recv_ap2ue2(AP2,UE,M8);claim_ue1(UE,Secret,RNue);claim_ue2(UE,Secret,AT);claim_ue3(UE,Secret,LAPID);claim_ue
4(UE,Secret,NUI);claim_
_ue5(UE,Secret,IMSI);claim_ue6(UE,Secret,NRES);claim_uea(UE,Alive);claim_uew(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(U
E,Nisynch);}role AP2{var RNws,RNue:Nonce;fresh RNap2:Nonce;secret SKap2ws:Function;const
LUI,AT,IMSI,LAPID,LNINLNI,KASME,RES,AKASuccess:Function;macro NUI=NUIFunc(LUI,RNue);macro
UNID=UNIDFunc(LNINLNI,RNws);macro NRES=KDF(RES,RNue,RNws);macro NMK=KDF(NKASME,RNap2);macro
M1={AT,RNue,IMSI,NUI,LAPID}pk(AP2);macro M2={AT,RNue,IMSI,LAPID,{NUI}sk(AP2)}SKap2ws;macro
M3={RNws,pk(UE),UNID}SKap2ws;macro M4={RNws,UNID}pk(UE);macro M5={NRES}sk(UE)}pk(AP2);macro
M6={RNap2,{NRES}sk(AP2)}SKap2ws;macro M7={NMK,AKASuccess}SKap2ws;macro
M8={AKASuccess,RNap2}pk(UE);recv_ueap21(UE,AP2,M1);send_ap2waaa1
(AP2,WAAA,M2);recv_waaaap21(WAAA,AP2,M3);send_ap2ue1(AP2,UE,M4);recv_ueap22(UE,AP2,M5);send_ap2waaa2(AP2,WAAA,M6);r
ecv_waaaap22(WAAA,AP2,M7);send_ap2ue2(AP2,UE,M8);claim_ap21(AP2,Secret,RNue);claim_ap22(AP2,Secret,AT);claim_ap23(AP2,Secre
t,LAPID);claim_ap24(AP2,Secret,NUI);claim_ap25(AP2,Secret,IMSI);claim_ap26(AP2,Secret,RNap2);claim_ap27(AP2,Secret,RNws);claim_a
p28(AP2,Secret,NRES);claim_ap29(AP2,Secret,AKASuccess);claim_ap210(AP2,Secret,UNID);claim_ap2a(AP2,Alive);claim_ap2w(AP2,Weak
agree);claim_ap2nia(AP2,Niagree);claim_ap2nis(AP2,Nisynch);}role WAAA{var RNap2,RNue:Nonce;
fresh RNws:Nonce;secret SKap2ws:Function;const LUI,AT,IMSI,LAPID,LNINLNI,KASME,RES,XRES,AKASuccess:Function;macro NUI=NUIFunc(LUI,RNue);macro
UNID= UNIDFunc(LNINLNI,RNws);macro NRES=KDF(RES,RNue,RNws);macro NXRES=KDF(XRES,RNue,RNws);macro
NKASME=KDF(KASME,RNws);macro NMK=KDF(NKASME,RNap2);macro M2={AT,RNue,IMSI,LAPID,{NUI}sk(AP2)}SKap2ws;macro
M3={RNws,pk(UE),UNID}SKap2ws;macro M6={RNap2,{NRES}sk(AP2)}SKap2ws;macro
M7={NMK,AKASuccess}SKap2ws;recv_ap2waaa1(AP2,WAAA,M2);send_waaaap21(WAAA,AP2,M3);recv_ap2waaa2(AP2,WAAA,M6);sen
d_waaaap22(WAAA,AP2,M7);claim_waaa1(WAAA,Secret,RNws);claim_waaa2(WAAA,Secret,UNID);claim_waaa3(WAAA,Secret,AKASucc
ess);claim_waaa4(WAAA,Secret,NMK);claim_waaa5(WAAA,Alive);claim_waaa6(WAAA,Weakagree);claim_waaa7(WAAA,Niagree);claim_
_waaa8(WAAA,Nisynch);}}

```

//Description of REPS-AKA4.4.1 Protocol

```

const pk:Function;secret sk:Function;inversekeys (pk,sk);hashfunction KDF,NUIFunc;protocol REPSAKA441(UE,AP3,WAAA2,LAAA){role
UE (var RNls:Nonce;fresh RNue: Nonce;const UI,AT,LAPID,WLID,IMSI,NINNI,RES,AKASuccess:Function;macro
NUI=NUIFunc(RNue,UI);macro UNINNI=NUIFunc(RNls,NINNI);macro NRES=KDF(RES, RNls,RNue);macro
MK=KDF(UNINNI,UI,NRES);macro M1={RNue,AT,LAPID,WLID,{NUI}sk(UE),IMSI}pk(AP3);macro M6={UNINNI,RNls}pk(UE);macro
M7={NRES}sk(UE)}
M10={AKASuccess}pk(UE);send_ueap31(UE,AP3,M1);recv_ap3ue1(AP3,UE,M6);send_ueap32(UE,AP3,M7);recv_ap3ue2(AP3,UE,M10);clai
m_ue1 (UE
,Secret,RNue);claim_ue2(UE,Secret,AT);claim_ue3(UE,Secret,LAPID);claim_ue4(UE,Secret,WLID);claim_ue5(UE,Secret,IMSI);claim_ue6(U
E,Secret,NUI);claim_ue7(UE,Secret,NRES);claim_uea(UE,Alive);claim_uew(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisync
h);role AP3{var RNue,RNls:Nonce;secret SKap3ws2 :Function;const UI,AT,LAPID,WLID,IMSI,NINNI,RES,AKASuccess:Function;macro
NUI=NUIFunc(RNue,UI);macro UNINNI=NUIFunc(RNls,NINNI);macro NRES=KDF(RES, RNls,RNue);macro
MK=KDF(UNINNI,UI,NRES);macro M1={RNue,AT,LAPID,WLID,{NUI}sk(UE),IMSI}pk(AP3);macro
M2={AT,NUI,IMSI,LAPID,RNue,WLID}SKap3ws2; macro M5={UNINNI,pk(UE),RNls}SKap3ws2;macro
M6={UNINNI,RNls}pk(UE);macro M7={NRES}sk(UE)}pk(AP3);macro M8={NRES}sk(UE)}SKap3ws2;macro M9={
AKASuccess,MK}SKap3ws2;macro
M10={AKASuccess}pk(UE);recv_ueap31(UE,AP3,M1);send_ap3waaa21(AP3,WAAA2,M2);recv_waaa2ap31(WAAA2,AP3,M5); send
_ap3ue1(AP3,UE,M6);recv_ueap32(UE,AP3,M7);send_ap3waaa22(AP3,WAAA2,M8);recv_waaa2ap32(WAAA2,AP3,M9);send_ap3ue2(AP3,
UE,M10);claim_ap31(AP3,Secret,RNue);claim_ap32(AP3,Secret,AT);claim_ap33(AP3,Secret,LAPID);claim_ap34(AP3,Secret,UNINNI);claim
_ap35(AP3,Secret,IMSI);claim_ap36(AP3,Secret,NUI);claim_ap37(AP3,Secret,WLID);claim_ap38(AP3,Secret,NRES);claim_ap39(AP3,Secret,
AKASuccess);claim_ap310(AP3,Secret,RNls);claim_ap3a(AP3,Alive);claim_ap3w(AP3,Weakagree);claim_ap3nia(AP3,Niagree);claim_ap3nis(
AP3,Nisynch);}role WAAA2{var RNue,RNls:Nonce;secret SKap3ws2,SKws2ls:Function;const UI,AT,LAPID,
WLID,IMSI,NINNI,RES,XRES,AKASuccess:Function;macro NUI=NUIFunc(RNue,UI);macro NXRES=KDF(XRES,RNls,RNue);macro
UNINNI=NUIFunc(RNls,NINNI);macro NRES=KDF(RES,RNls,RNue);macro MK=KDF(UNINNI,UI,NRES);macro
M2={AT,NUI,IMSI,LAPID,RNue,WLID}SKap3ws2;macro M3={RNue,WLID,IMSI,NUI,AT}SKws2ls; macro
M4={NXRES,{UNINNI}sk(LAAA),pk(UE),RNls}SKws2ls;macro M5={UNINNI,pk(UE),RNls}SKap3ws2;macro
M8={NRES}sk(UE)}SKap3ws2;macro M9={
AKASuccess,
MK}SKap3ws2;recv_ap3waaa21(AP3,WAAA2,M2);send_waaa2laaa1(WAAA2,LAAA,M3);recv_laaawaaa21(LAAA,WAAA2,M4);send_waaa
2ap31(WAAA2,AP3,M5);
recv_ap3waaa22(AP3,WAAA2,M8);send_waaa2ap32(WAAA2,AP3,M9);claim_waaa21(WAAA2,Secret,RNue);claim_waaa22(WAAA2,Secret,
WLID);claim_waaa23(WAAA2,Secret,IMSI);claim_waaa24(WAAA2,Secret,NUI);claim_waaa25(WAAA2,Secret,AT);claim_waaa26(WAAA2,
Secret,UNINNI);claim_waaa27(WAAA2,Secret,RNls);
claim_waaa28(WAAA2,Secret,AKASuccess);claim_waaa2a(WAAA2,Alive);claim_waaa2w(WAAA2,Weakagree);claim_waaa2nia(WAAA2,Ni
agree);claim_waaa2nis (WAAA2,Nisynch);}role LAAA{var RNue:Nonce;secret SKws2ls:Function;const
UI,AT,LAPID,WLID,IMSI,NINNI,XRES,AKASuccess:Function;macro
NUI=NUIFunc(RNue,UI);macro
UNINNI=NUIFunc(RNls,NINNI);macro NXRES=KDF(XRES,RNls,RNue);macro M3={RNue,WLID,IMSI,NUI,AT}SKws2ls;macro
M4={NXRES
,{UNINNI}sk(LAAA),pk(UE),RNls}SKws2ls;recv_waaa2laaa1(WAAA2,LAAA,M3);send_laaawaaa21(LAAA,WAAA2,M4);claim_laaa1(LA
AA,Secret,NXRES);claim_laaa2(LAAA,Secret,UNINNI);claim_laaa3(LAAA,Secret,RNls);claim_laaa4(LAAA,Alive);claim_laaaw(LAAA,Wea
kagree);claim_laaania(LAAA,Niagree);claim_laaanis(LAAA,Nisynch);}

```

//Description of REPS-AKA4.2 Protocol

```

const pk:Function;secret sk:Function;inversekeys (pk,sk);hashfunction KDF,NUIFunc,UNIDFunc;protocol
REPSAKA442(UE,AP3,WAAA2,LAAA){role UE (var RNls:Nonce; fresh RNue:Nonce;const
LUI,AT,IMSI,LAPID,WLID,LNINLNI,RES,AKASuccess:Function;macro NUI=NUIFunc(LUI,RNue);macro
ULNINLNI=UNIDFunc(LNINLNI,RNls); macro NRES=KDF(RES,RNls,RNue);macro MK=KDF(ULNINLNI,NUI,NRES);macro M1={
AT,RNue,IMSI,NUI,LAPID,WLID}pk(AP3);macro M6={RNls,ULNINLNI}pk(UE); macro M7={NRES}sk(UE)}pk(AP3);macro
M10={AKASuccess}pk(UE);send_ueap31(UE,AP3,M1);recv_ap3ue1(AP3,UE,M6);send_ueap32(UE,AP3,M7); recv_ap3ue2
(AP3,UE,M10);claim_ue1(UE,Secret,RNue);claim_ue2(UE,Secret,AT);claim_ue3(UE,Secret,LAPID);claim_ue4(UE,Secret,WLID);claim_ue5(
UE,Secret,IMSI);claim_ue6(UE,Secret, NUI);claim_ue7(UE,Secret,NRES);claim_uea (UE,Alive);claim_uew
(UE,Weakagree);claim_uenia(UE,Niagree);claim_uenis(UE,Nisynch);}role AP3{var RNue,RNls:Nonce;secret SKap3ws2:Function;const
LUI,AT,IMSI,LAPID,WLID,LNINLNI,RES,AKASuccess:Function;macro NUI=NUIFunc(LUI,RNue);macro ULNINLNI=
UNIDFunc(LNINLNI,RNls);macro NRES=KDF(RES,RNls,RNue);macro MK=KDF(ULNINLNI,NUI,NRES);macro M1={
AT,RNue,IMSI,NUI,LAPID,WLID}pk(AP3);macro M2={ AT,RNue,IMSI, NUI,WLID,LAPID}SKap3ws2;macro
M5={RNls,pk(UE),ULNINLNI}SKap3ws2;macro M6={RNls,ULNINLNI}pk(UE);macro M7={NRES}sk(UE)}pk(AP3); macro
M8={NRES}SKap3ws2;macro M9={MK,AKASuccess}SKap3ws2;macro
M10={AKASuccess}pk(UE);recv_ueap31(UE,AP3,M1);send_ap3waaa21(AP3,WAAA2,M2);
recv_waaa2ap31(WAAA2,AP3,M5);send_ap3ue1(AP3,UE,M6);recv_ueap32(UE,AP3,M7);send_ap3waaa22(AP3,WAAA2,M8);recv_waaa2ap3
2(WAAA2,AP3,M9);send_ap3ue2(AP3,UE, M10);claim_ap31 (AP3, Secret, RNue);claim_ap32 (AP3, Secret, AT); claim_ap33 (AP3, Secret,
LAPID); claim_ap34 (AP3, Secret,ULNINLNI); claim_ap35 (AP3, Secret, IMSI); claim_ap36 (AP3, Secret, NUI); claim_ap37 (AP3, Secret,
WLID); claim_ap38 (AP3, Secret, NRES);claim_ap39 (AP3, Secret, AKASuccess);claim_ap310 (AP3, Secret, RNls); claim_ap3a (AP3, Alive);
claim_ap3w (AP3, Weakagree); claim_ap3nia (AP3, Niagree);claim_ap3nis (AP3, Nisynch);}role WAAA2{var RNue, RNls:Nonce;secret
SKap3ws2, SKws2ls :Function;const LUI, AT, IMSI,LAPID,WLID, LNINLNI,RES,XRES,AKASuccess :Function;macro NUI= NUIFunc(LUI,
RNue);macro NXRES=KDF(XRES,RNls, RNue);macro NRES=KDF(RES,RNls, RNue);macro ULNINLNI=UNIDFunc(LNINLNI,RNls);macro
MK=KDF(ULNINLNI ,NUI, NRES);macro M2={ AT,RNue, IMSI, NUI, WLID,LAPID}SKap3ws2; macro M3={
AT,RNue,IMSI,NUI,WLID,LAPID}SKws2ls;macro M4={RNls, pk(UE),{ULNINLNI} sk(LAAA, NXRES)}SKws2ls;macro M5={
RNls,pk(UE),ULNINLNI}SKap3ws2; macro M8={ NRES}SKap3ws2; macro M9={ MK,AKASuccess}SKap3ws2; recv_ap3waaa21
(AP3,WAAA2, M2);send_waaa2laaa1(WAAA2,LAAA, M3);recv_laaawaaa21(LAAA,WAAA2, M4);send_waaa2ap31(WAAA2,AP3,
M5);recv_ap3waaa22(AP3,WAAA2, M8);send_waaa2ap32(WAAA2,AP3, M9);claim_waaa21(WAAA2,Secret,RNue);claim_waaa22 (WAAA2,
Secret, WLID); claim_waaa23 (WAAA2, Secret, IMSI); claim_waaa24 (WAAA2, Secret, NUI);claim_waaa25(WAAA2, Secret, AT);
claim_waaa26 (WAAA2,Secret,ULNINLNI); claim_waaa27 (WAAA2, Secret, RNls); claim_waaa28 (WAAA2, Secret,
AKASuccess);claim_waaa29 (WAAA2, Secret, LAPID); claim_waaa210 (WAAA2, Secret, MK);claim_waaa2a (WAAA2, Alive); claim_waaa2w
(WAAA2, Weakagree); claim_waaa2nia (WAAA2, Niagree);claim_waaa2nis (WAAA2, Nisynch);}role LAAA{var RNue:Nonce;fresh RNls:
Nonce; secret SKws2ls :Function;const LUI, AT, IMSI,LAPID,WLID, LNINLNI,XRES,AKASuccess :Function;macro NUI=NUIFunc(LUI,
RNue);macro ULNINLNI=UNIDFunc(LNINLNI, RNls);macro NXRES=KDF(XRES,RNls, RNue);macro M3={
AT,RNue,IMSI,NUI,WLID,LAPID}SKws2ls;macro M4={ RNls, pk(UE),{ULNINLNI}sk(LAAA), NXRES}SKws2ls;
recv_waaa2laaa1(WAAA2,LAAA, M3);send_laaawaaa21(LAAA,WAAA2, M4);claim_laaa1 (LAAA, Secret, NXRES); claim_laaa2 (LAAA,
Secret, ULNINLNI); claim_laaa3 (LAAA, Secret, RNls);claim_laaa4 (LAAA, Alive); claim_laaaw (LAAA, Weakagree); claim_laaania (LAAA,
Niagree);claim_laaanis (LAAA, Nisynch);}

```

//Description of REPS-AKA5 Protocol

```

const pk:Function; secret sk:Function;inversekeys (pk,sk);usertype Key;hashfunction
NIFunc,F1,F2,F3,F4,F5,F6,F7,F8,AUTHFunc,F9,KDF,XOR;protocol REPSAKA5(IOTD ,IOTL,MME,HSS){role IOTD{var
RNgm,RNmm,RNhs,Tm:Nonce;fresh RNnm:Nonce;onst IMSI,GIKm, DCID1,DCIDROG,GXKm,DCIDnm,GIDm,SKmh,SKnm:Function;
macro DTIDnm=F1(RNnm,IMSI,GIKm);macro SIGm=F1(RNgm,XOR(DCID1,DCIDROG),GXKm,DCIDnm,GIDm,SKmh,SKnm);macro
MK=KDF(XOR(RNhs,RNgm),GXKm);macro AUTHM=F8(SIGm, XOR (RNgm,RNhs),MK);macro
XRESnm=F6(DCIDnm,RNhs,SKnm);macro SKDLnm=KDF(XOR(RNgm,RNnm),GIKm);macro AUTHnm=F9(AUTHM,SKDLnm);macro
UPK=KDF (XOR(RNmm,RNgm),SIGm,MK);macro NAUTHM=F8(XOR(RNmm,RNgm),AUTHM,UPK);macro
NSKDLnm=KDF(XOR(RNgm,RNnm),SKDLnm,GIDm);macro NAUTHnm= F9(NAUTHM,NSKDLnm);macro

```

```

NDInm=F1(XOR(AUTHnm,RNnm),SKDLnm);macro TDInm=F1(XOR(IMSI,RNnm),GIKnm);macro TSKDLnm=KDF(XOR (RNgm,RNnm),NAUTHm,GIKnm);macro TAUTHnm=F8(NAUTHm,XOR(RNgm,RNnm),TSKDLnm);macro
M1nm={RNnm,{DTIDnm}sk(IOTD)}pk(IOTL);macro M6nm={{AUTHm}sk(IOTL),RNgm,RNhss}pk(IOTD);macro
M7nm={{XRESnm}sk(IOTD),AUTHnm}pk(IOTL);macro CM1nm={DTIDnm,{RNnm}sk(IOTD)}pk(IOTL);macro CM4nm={{NAUTHm} sk (IOTL),RNgm}pk(IOTD);macro CM5nm={{NAUTHnm}sk(IOTD)}pk(IOTL);macro IM1nm={DCIDnm,RNnm,{NDInm}sk(IOTD)}pk(IOTL);macro IM2nm={{NAUTHm} sk(IOTL),RNgm}pk(IOTD);macro
IM3nm={{NAUTHnm}sk(IOTD),RNgm}pk(IOTL);macro UM1nm={IMSI,RNnm,{TDInm}sk(IOTD)}pk(IOTL);macro
UM2nm={{NAUTHm}sk(IOTL),RNgm}pk(IOTD);macro
UM3nm={{TAUTHnm}sk(IOTD),RNgm}pk(IOTL);send_iotdiot1(IOTD,IOTL,M1nm);recv_iotdiot1(IOTL,IOTD,M6nm);send_iotdiot2(IOTD,IOTL,M7nm);send_iotdiot3(IOTD,IOTL,CM1nm);recv_iotdiot2(IOTL,IOTD,CM4nm);send_iotdiot4(IOTD,IOTL,CM5nm);send_iotdiot5(IOTD,IOTL,IM1nm);recv_iotdiot3(IOTL,IOTD,IM2nm);send_iotdiot6(IOTD,IOTL,IM3nm);send_iotdiot7(IOTD,IOTL,UM1nm);recv_iotdiot4(IOTL,IOTD,UM2nm);send_iotdiot8(IOTD,IOTL,UM3nm); claim_iotd1 (IOTD,Secret,RNnm); claim_iotd2 (IOTD,Secret,DTIDnm); claim_iotd3 (IOTD,Secret,XRESnm); claim_iotd4 (IOTD,Secret,AUTHnm);claim_iotd5 (IOTD,Secret,NDInm); claim_iotd6 (IOTD,Secret,NAUTHnm); claim_iotd7 (IOTD,Secret,IMSI); claim_iotd8 (IOTD,Secret,TDInm);claim_iotd9 (IOTD,Secret,TAUTHnm);claim_iotda (IOTD,Alive); claim_iotdw (IOTD,Weakagree); claim_iotdnia (IOTD,Niagree); claim_iotdnis (IOTD,Nisynch);}role IOTL{var RNnm,RNmme, RNhss:Nonce;fresh RNgm,Tm:Nonce;const IMSI,DCID1,DCIDROG,GXKm,GIDm,GIKnm,MVP,HVP,SQL,DCIDnm,XRESnmROG,SKmh,SKnm:Function;macro DTIDnm=F1(RNnm,IMSI,GIKnm);macro SIGm=F1(RNgm,XOR(DCID1,DCIDROG),GXKm);macro HNID=NIFunc(XOR(MVP,HVP));macro MK=KDF(XOR(RNhss,RNgm),GXKm);macro CK=F3(RNhss,MK);macro IK=F4(RNhss,MK);macro AK=F5(RNhss,MK);macro KASME=KDF(XOR(SQN,AK),HNID,CK,IK,MK);macro KASMEm=KDF(RNhss,KASME,SIGm,GXKm);macro AUTHm=F8(SIGm,XOR(RNgm,RNhss),MK);macro SKDLnm=KDF(XOR(RNgm,RNnm),GIKnm);macro XRESnm=F6(DCIDnm,RNhss,SKnm); macro AUTHnm=F9(AUTHm,SKDLnm); macro MXRES=F2(XOR(RNhss,RNgm),GXKm);macro XRESlm=F7(MXRES,XOR(XRESnm,XRESnmROG));macro UHNI=AUTHFunc(XOR(RNgm,RNhss),AUTHm);macro UPK=KDF(XOR(RNmme,RNgm),SIGm,MK);macro NAUTHm=F8(XOR(RNmme,RNgm),AUTHm,UPK);macro NKASMEm=KDF(XOR(RNmme,RNgm),KASMEm,UPK);macro NSKDLnm=KDF(XOR(RNgm,RNnm),SKDLnm,GIDm);macro NAUTHnm=F9(NAUTHm,NSKDLnm);macro NXRESlm=F7(XOR(RNmme,RNgm),XRESlm,UPK);macro NUHNI=AUTHFunc(XOR(RNmme,RNgm),UHNI,UPK);macro NDInm=F1(XOR(AUTHnm,RNnm),SKDLnm);macro TDInm=F1(XOR(IMSI,RNnm),GIKnm);macro TSKDLnm=KDF(XOR(RNgm,RNnm),NAUTHm,GIKnm);macro TAUTHnm=F8(NAUTHm,XOR(RNgm,RNnm),TSKDLnm);macro M1nm={RNnm,{DTIDnm}sk(IOTD)}pk(IOTL);macro M2nm={GIDm,{SIGm}sk(IOTL),Tm,RNgm}pk(MME);macro M5m={{AUTHm}sk(MME),RNhss,HNID}pk(IOTL);macro M6nm={{AUTHm}sk(IOTL),RNgm,RNhss}pk(IOTD);macro M7nm={{XRESnm}sk(IOTD),AUTHnm}pk(IOTL);macro M8nm={{XRESlm}sk(IOTL)}pk(MME);macro CM1nm={DTIDnm,{RNnm}sk(IOTL),RNgm,RNhss}pk(IOTD);macro CM2m={GIDm,Tm,RNgm,{UHNI}sk(IOTL)}pk(MME);macro CM3m={{NAUTHm}sk(MME),RNnmme}pk(IOTL);macro CM4nm={{NAUTHm} sk (IOTL),RNgm}pk(IOTD);macro CM5nm={{NAUTHnm}sk(IOTD)}pk(IOTL);macro CM6m={{NXRESlm}sk(IOTL)}pk(MME);macro CM7m={{UHNI}sk(IOTL)}pk(MME);macro XM2m={{NAUTHm}sk(MME),RNnmme}pk(IOTL);macro XM3m={{NXRESlm}sk(IOTL),RNnmme}pk(MME);macro IM1nm={DCIDnm,RNnm,{NDInm}sk (IOTD)}pk(IOTL);macro IM2nm={{NAUTHm}sk(IOTL),RNgm}pk(IOTD);macro IM3nm={{NAUTHnm}sk(IOTD),RNgm}pk(IOTL);macro UM1nm={IMSI,RNnm,{TDInm}sk (IOTD)}pk(IOTL);macro UM2nm={{NAUTHm}sk(IOTL),RNgm}pk(IOTD);macro UM3nm={{TAUTHnm}sk(IOTD),RNgm}pk(IOTL);recv_iotdiot1(IOTD,IOTL,M1nm); send_iotlmm1(IOTL,MME,M2m);recv_mmeiot1(MME,IOTL,M5m);send_iotlmiot1(IOTL,IOTD,M6nm);recv_iotdiot2(IOTD,IOTL,M7nm);send_iotlmm2(IOTL,MME,M8m);recv_iotlmiot2(IOTD,IOTL,CM1nm);send_iotlmm3(IOTL,MME,CM2m);recv_mmeiot2(MME,IOTL,CM3m);send_iotlmiot2(IOTL,IOTD,CM4nm);recv_iotdiot4 (IOTD,IOTL,CM5nm);send_iotlmm4(IOTL,MME,CM6m);send_iotlmm5(IOTL,MME,XM1m);recv_mmeiot3(MME,IOTL,XM2m);send_iotlmm6(IOTL,MME,XM3m);recv_iotdiot5(IOTD,IOTL,IM1nm);send_iotlmiot3(IOTL,IOTD,IM2nm);recv_iotdiot6(IOTD,IOTL,IM3nm);recv_iotdiot7(IOTD,IOTL,UM1nm); send _iotdiot4 (IOTL ,IOTD,UM2nm);recv_iotdiot8(IOTD,IOTL,UM3nm);claim_iotl1 (IOTL,Secret,GIDm); claim_iotl2 (IOTL,Secret,SIGm); claim_iotl3 (IOTL,Secret,Tm); claim_iotl4 (IOTL,Secret,RNgm);claim_iotl5 (IOTL,Secret,AUTHm); claim_iotl6 (IOTL,Secret,RNhss); claim_iotl7 (IOTL,Secret,XRESlm); claim_iotl8 (IOTL,Secret,UHNI);claim_iotl9 (IOTL,Secret,NAUTHm); claim_iotl10 (IOTL,Secret,NXRESlm);claim_iotla (IOTL,Alive); claim_iotlw (IOTL,Weakagree); claim_iotlnia (IOTL,Niagree); claim_iotlnis (IOTL,Nisynch);}role MME{var RNnm,RNgm,RNhss,Tm:Nonce;fresh RNnmme:Nonce;const DCID1,DCIDROG,GXKm,GIDm,IMSI,MVP,HVP ,XRESnmROG,DCIDnm,SQL,SKmh,SKnm:Function;macro SIGm=F1(RNgm,XOR(DCID1,DCIDROG),GXKm);macro HNID=NIFunc(XOR(MVP,HVP));macro MK=KDF(XOR(RNhss,RNgm),GXKm); macro MXRES=F2(XOR(RNhss,RNgm),GXKm);macro CK=F3(RNhss,MK);macro IK=F4(RNhss,MK);macro AK=F5(RNhss,MK);macro XRESnm=F6(DCIDnm,RNhss,SKnm);macro XRESlm=F7(MXRES,XOR(XRESnm,XRESnmROG));macro KASME=KDF(RNhss,KASME,SIGm,GXKm);macro AUTHm=F8(SIGm,XOR(RNgm,RNhss),MK);macro UHNI=AUTHFunc(XOR(RNgm,RNhss),AUTHm);macro NKASMEm=KDF(XOR(RNmme,RNgm),KASMEm,UPK);macro NAUTHm=F8(XOR(RNmme,RNgm),AUTHm,UPK);macro NXRESlm=F7(XOR(RNmme,RNgm),XRESlm,UPK);macro M2m={GIDm,{SIGm}sk(IOTL),Tm,RNgm}pk(MME);macro M3m={GIDm,{HNID}sk(MME),RNgm,SIGm}SKmh;macro M4m={XRESlm,{AUTHm}sk(HSS),KASMEm,RNhss}SKmh;macro M5m={{AUTHm}sk(MME),RNhss,HNID}pk(IOTL);macro M8m={{XRESlm}sk(IOTL)}pk(MME);macro CM2m={GIDm,Tm,RNgm,{UHNI}sk(IOTL)}pk(MME);macro CM3m={{NAUTHm}sk(MME),RNnmme}pk(IOTL);macro XM2m={{NAUTHm}sk(MME),RNnmme}pk(IOTL);macro XM3m={{NXRESlm}sk(IOTL),RNnmme}pk(MME);recv_iotlmm1(IOTL,MME,M2m); send_mmehss1(MME,HSS,M3m);recv_hssmme1(HSS,MME,M4m);send_mmeiot1(MME,IOTL,M5m);recv_iotlmm2(IOTL,MME,M8m);recv_iotlmm3(IOTL,MME,CM2m);send_mmeiot2(MME,IOTL,CM3m);recv_iotlmm4(IOTL,MME,CM6m);recv_iotlmm5 (IOTL, MME, XM1m);send_mmeiot3(MME,IOTL,XM2m);recv_iotlmm6(IOTL,MME,XM3m);claim_mme1 (MME,Secret,GIDm); claim_mme2 (MME,Secret,HNID); claim_mme3 (MME,Secret,RNgm); claim_mme4 (MME,Secret,SIGm); claim_mme5 (MME,Secret,AUTHm); claim_mme6 (MME,Secret,RNhss); claim_mme7 (MME,Secret,NAUTHm); claim_mme8 (MME,Secret,RNmme); claim_mmea (MME,Alive); claim_mmeew (MME,Weakagree); claim_mmenia (MME,Niagree);claim_mmenis (MME,Nisynch);}role HSS{fresh RNhss:Nonce;var RNnm,RNgm,RNmme ,Tm:Nonce;const GIDm,IMSI,MVP,HVP, GXKm, SQN,AMF,DCID1,DCIDROG,DCIDnm, SKnm,XRESnmROG,SKmh,SKnm:Function;macro HNID=NIFunc(XOR(MVP,HVP));macro SIGm=F1(RNgm,XOR(DCID1,DCIDROG),GXKm);macro MK=KDF(XOR (RNhss,RNgm),GXKm);macro MXRES=F2(XOR(RNhss,RNgm),GXKm);macro CK=F3(RNhss,MK);macro IK=F4(RNhss,MK);macro AK=F5(RNhss,MK);macro XRESnm=F6(DCIDnm,RNhss,SKnm);macro XRESlm=F7(MXRES,XOR(XRESnm,XRESnmROG));macro KASME=KDF(XOR(SQN,AK),HNID,CK,IK,MK);macro KASMEm=KDF(RNhss,KASME,SIGm,GXKm);macro AUTHm=F8(SIGm,XOR(RNgm,RNhss),MK);macro M3m={GIDm,{HNID}sk(MME),RNgm,SIGm}SKmh;macro M4m={{XRESlm,{AUTHm}sk(HSS), KASMEm, RNhss}SKmh;recv_mmehss1(MME,HSS,M3m);send_hssmme1(HSS,MME,M4m);claim_hss1 (HSS,Secret,XRESlm); claim_hss2 (HSS,Secret,AUTHm); claim_hss3 (HSS,Secret,KASMEm); claim_hss4 (HSS,Secret,RNhss);claim_hssa (HSS,Alive); claim_hssw (HSS,Weakagree); claim_hssnia (HSS,Niagree);claim_hssnis (HSS,Nisynch);}

```