Investigation of the Method of Authenticated Key Exchange

Cyril Chinonye Ede

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

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
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 Master of Science in Computer Engineering.

Assoc. Prof. Dr. Alexander Chefranov
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Alexander Chefranov
2. Assoc. Prof. Dr. Gürcü Öz
3. Asst. Prof. Dr. Önsen Toygar

ABSTRACT

This thesis work investigates the method of authenticated key exchange, a key exchange protocol where communicating parties generate and exchange secret session keys for the purpose of authentication. We focused attention on the Efficient Two-Server Password-Only Authenticated Key Exchange by Xun Yi, San Ling, and Huaxiong Wang, the most recent authenticated key exchange protocol. The protocol uses two-server scenario to offer a symmetric solution for authenticated key exchange protocol in the password-only model. Before authentication process, the client chooses a password and computes password authenticator such that the password cannot be revealed from the authenticator to anyone, except the two servers conspired and sends to the two servers through a secure channel. The protocol generates shared session keys by each communicating party such that, in the result of their computations, parties arrive at a common session key. In our investigation, we discovered a problem with the protocol that will cause its failure and render it inefficient in general scenario. The problem is that the protocol does not take into account congruency of the exponents modulo Euler's totient function, resulting in the parties arriving at different session keys at the end of computations, which we proved and illustrated by numerical counter-example. We provided a modification to the protocol by proposing that in choosing parameters whose inverses are involved in computations, care must be taken to ensure that their multiplicative inverses modulo Euler's totient function exist. We provided a proof for the modification and a numerical example to illustrate the correctness of this modification and confirmed

that the protocol works efficiently. With the proposed modification, it is certain the protocol will function without any failure.

Keywords: Authenticated key exchange protocol, two-server architecture, password-only authentication, Diffie-Hellman, ElGamal encryption scheme

ÖZ

Bu çalışma, kimlik doğrulama amacıyla, iletişim içerisinde bulunan tarafların gizli oturum anahtarları oluşturup değiştirdiği bir kimlik anahtarı protokolü olan, kimliği doğrulanan anahtar değişimi yöntemini araştırmaktadır. Xun Yi, San Ling ve Huaxiong Wang tarafından geliştirilen ve en güncel kimliği doğrulanan anahtar değişimi protokolü olan Etkili İki-Sunuculu Yalnızca-Parolalı Kimliği Doğrulanan Anahtar Değişimi üzerine odaklanılmıştır. Yalnızca parola modelindeki kimliği doğrulanan anahtar değişimi protokolü için bir simetrik çözüm sunmak üzere protokol tarafından iki sunuculu bir örnek kullanılmaktadır. İstemci, kimlik doğrulama sürecinden önce bir parola seçer; ardından parola kimlik doğrulayıcısını, parolanın ilgili iki sunucu dışında gösterilmeyeceği şekilde hesaplar ve parolayı iki sunucuya güvenli bir kanal aracılığıyla gönderir. Protokol, her iletişim kuran taraf ile ortak oturum anahtarları oluşturur ve hesaplamaların bir sonucu olarak taraflar ortak bir anahtarda karar kılar. Araştırmamızda, protokol ile ilgili bir hataya rastlanmış ve genel olarak yetersiz bir şekilde işlenmesine neden olabilecek bir sorun keşfedilmiştir. Sorun, protokolün Euler totient fonksiyonu üs modulosunun eşleşikliğini değerlendirmemesi ve bunun bir sonucu olarak hesap sonlarında tarafların farklı oturum anahtarlarında bulunmalarıdır. Bu sorun sayısal örneklerle gösterilmiştir ve ispatlanmıştır. Hesaplamalarda tersleri bulunan parametreler seçerek protokole bir değişiklik sağlanmıştır. Bu durumda, çarpımsal tersler modulosu Euler toteint fonksiyonunun mevcut olduğu konusunda dikkatli olunmalıdır. Değişikliğin doğruluğunu gösteren bir kanıt ile sayısal bir örnek sunulmuştur ve protokolün etkili

bir biçimde çalıştığı anlaşılmıştır. Önerilen değişiklik ile protokolün hata ile karşılaşmadan çalışacağı belirlenmiştir.

Anahtar Sözcükler: Kimliği doğrulanan anahtar değişimi protokolü, iki sunuculu yapı, yalnızca parolalı kimlik doğrulama, Diffie-Hellman, ElGamal kriptolama düzeni.

Dedicated to my father, Mr. Silas E. Odo, whose love for me knows no bounds.

ACKNOWLEDGMENT

I have witnessed the end of a superior academic tussle, and it is indeed an exceptional measure in the direction of gigantic academic standing. I owe copious gratitude to God Almighty for His protection and unequal favor. I am obliged to my supervisor, Prof. Dr. Alexander Chefranov, for all his efforts to leading me to the field of research in Network Security. You are instrumental in my academic development. I appreciate my Thesis defence Jury Chair, Asst. Prof. Dr. Gürcü Öz and Jury member, Asst. Prof. Dr. Önsen Toygar for their efforts during my defence. In a peculiar way, I am indebted to Prof. Hilary O. Edeoga, the Vice Chancellor of the Michael Okpara University of Agriculture Umudike, whom God has used to resurrect the dry bones in me. This work would be incomplete without expressing heartfelt appreciation to my father, Mr. Silas E. Odo, whose love for me knows no bounds. He always finds a way to put my legs down when in turmoil. I wish to appreciate my loving siblings: Rita, Austin (the Mathematician), Ezekiel (the Physicist) and the baby of the house, Victoria. I would love to have you people as siblings in my next world, if it will exist. I am indebted to Miss Asadu Lilian for all her understanding and encouragements throughout this program. I owe appreciation to colleagues in the Department of Computer Science, all the staff of the Michael Okpara University of Agriculture, Umudike and ETF for all their supports. I remain indebted to all my friends, Blessing Chioma Okoye, Collins, Fidel, and Tope, especially colleagues in the struggle and all who contributed to the accomplishment of this great academic expedition. I appreciate you Nnaemeka Nnabude for being a wonderful roommate. May the Almighty God reward you all abundantly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AKE	Authenticated Key Exchange
IEEE	Institute of Electrical and Electronics Engineers
PAKE	Password Authenticated Key Exchange
PKI	Public Key Infrastructure
Req	Request
SSL	Secure Socket Layer
YLW	Yi-Ling-Wang

Chapter 1

INTRODUCTION

1.1 Background of the Study

Password is the most used means to access secure systems such as email servers, computer operating systems, mobile phones, automated teller machines, etc. It does not cost anything for a user to think out a password to enable him or her access a secure system. This password could be any memorable word or string of characters coined from anything the user can remember easily. However, due to the problem of remembrance, users choose a password with very low entropy, thus making it susceptible to brute-force dictionary attacks.

In a password authenticated key exchange (PAKE), client and server that share password, authenticate each other using the password, and arrive at same key. As a user inputs his or her password to access a secure system, the hash value of that password transmits through an insecure channel to the server for authentication, thus exposing it to possible adversary's activities. The above scenarios are what happen in a typical protocol for password-based authentication system, where a single server stores the whole password for the client's authentication. This protocol is a weak system because when an adversary compromises the server; the adversary's activities reveal all the stored passwords to the attacker.

Two-server password-based authentication protocol was presented by [1], [2], [3], [4] and [5] to avert this vulnerability issue described above. Two-server password-based authentication is a protocol that allows two servers collaborate in verifying the identity of a client. In this two-server architecture, the servers do not need to store or have the knowledge of the client's password. The client sends authentication information, based on the chosen password, to the servers. In this system, if the adversary attacks one of the servers, it will not be possible to fool the other server to be the client. This two-server architecture operates in either asymmetric or symmetric mode. In asymmetric, one server supports the other in the authentication process while, in symmetric, the both servers co-operate to authenticate the client.

Yi, Ling and Wang [5] offered a symmetric solution (YLW scheme) intended for two-server password-only authenticated key exchange that is only practical symmetric solution. The protocol is such that the two servers cooperate to authenticate a client, and when adversary attacks one server, it will not be possible to fool the other server for being the client. The protocol runs in three phases of which at each, the communicating parties choose some parameters at random from a list of all invertible elements in a generated cyclic group of large prime number. At the initialization phase, the two servers generate a cyclic group based on a big prime number whose generator is q, and a safe hash function. They make these parameters public, which the communicating parties use throughout the rest of the computations in the protocol to arrive at same keys. During the second phase, the registration phase, the client, for the purpose of encryption and decryption, generates pairs of keys. It selects a password and uses the encryption key to encrypt it according to the

ElGamal encryption scheme [6]. In the last stage, the authentication and key exchange phase, the parties arrive at same secret keys at the end of computations.

This work describes the problem with the YLW protocol and proposes its modification allowing it to work correctly. The computations to arrive at same secret keys by the communicating parties are based on exponents congruent modulo *q*. In a case like this, it is known that the powers should be congruent modulo *q-1*. This event of congruency of powers modulo *q-1* is not considered in the protocol resulting in different keys that is proved by a numerical counter-example. Improper choosing of session keys or failure to establish shared session keys is the primary design issue in any given cryptosystem. We finally proposed a modification to the protocol to enable it work correctly, and illustrated this by numerical example.

1.2 Objective of the Thesis

This research will investigate the method of authenticated key exchange protocol presented by Xun Yi et al [5]. It will delve into all the computations to arrive at same secret session keys in the protocol and verify the correctness claim of the proof. The investigation is essential as any failure to arrive at same secret session keys, or improper choosing of these keys may lead to a flaw in the cryptosystem – authentication may fail.

This work will describe the problem discovered in the protocol by employing established facts. We will provide numerical counter-example to prove the actuality of such problem, where communicating parties at the end of computations arrive at different secret session keys. It will propose a modification to the protocol to

overcome this failure to come at same secret session keys and provide a numerical prove of the proposed modification.

1.3 Significance of the Study

In any cryptosystem, the central design issue is the failure to choose or arrive at proper secret session keys. This situation may leave the system susceptible to attack or system may crash. This research will provide information on the issue of two-server password-only authenticated key exchange. Further, it will review the problem with [5] protocol, which is because of, not considering congruency of exponents modulo Euler's totient and propose a solution to fix the problem. This modification will enable communicating parties in the protocol establish same secret session keys and exchange messages efficiently.

This study would be beneficial to organizations that would like to migrate from the use of one-server architecture to two-server architecture in their authentication processes to overcome the issue with one-server architecture. It will provide optimum secure communication for users seeking access to secure systems. It will help fill the research gap in [5] by proposing a modification that will enable communicating parties always arrive at same secret session keys at the end of computations in the protocol. To future researchers, this study can provide insight to the implementation and cryptanalysis of the protocol.

1.4 Organization of the Work

We organized the rest of this research work as follows: In Chapter 2, we discussed the concept of secure communication, basic password authentication key exchange, two-server authentication key exchange and gave a review of Diffie-Hellman key

exchange scheme [7], and ElGamal encryption protocol [6]. A detailed review of the YLW protocol [5] in question was provided, which is essential to understanding the prevailing problem with the protocol. Chapter 3 presented the problem with the protocol [5] using established facts and provided a proof and numerical counter-example to illustrate the protocol's failure. In Chapter 4, we proposed a modification to [5] and illustrated the correctness of the amendment with a numerical example. In Chapter 5, we provided a conclusion and recommended future work.

Chapter 2

LITERATURE REVIEW

2.1 Concept of Secure Communication

When server A communicates with server B, they do not want a third party, maybe server C to listen in. To ensure server C does not listen in or intercept the content of their communication, they need to communicate in a secure way. The secure way could be achieved either by hiding the content of their communication (using encryption, steganography). It could also be realized by hiding the communicating parties (anonymity), or by hiding the fact that communication takes place (security by obscurity). Secure communication ensues when communicating parties establish shared secret key with which they use to hide the contents of their communication, make themselves anonymous, or obscure their communication.

Secure communication is becoming commonplace in computer network system; between web sites and web browsers (SSL), distributed system [8]. As most communications take place over a distance through insecure channels, secure communication ensures user access, provides confidentiality, authentication, integrity and non-repudiation. Secure communication starts when the communicating parties exchange messages and establish a collective identity, called key, between them to ensure that messages are actually coming from the known system. The communicating parties use this key to encrypt and decrypt messages from each side

to avoid eavesdropping. A method for the exchange of keys to initiate secure communication is what this thesis is investigating.

2.1.1 Basic Password Authenticated Key Exchange

The less expensive and mostly used authentication mechanism in security applications is the password. Some authentication mechanisms such as the biometrics requires additional hardware resources that may be considered too costly for security application [9]. Due to the low entropy nature of the passwords, they need protection from transmission over insecure channels. The means of protecting these passwords, is by encryption, translating them into unreadable strings such that it makes no sense to any adversary.

The essence of authenticated key exchange (AKE) is for two communicating parties to arrive at shared key used in protecting subsequent communication on an insecure channel after identifying each other. On the other hand, in password-based key exchange (PAKE), two communicating parties can authenticate themselves using the password and arrive at a common secret session key for subsequent communication over an insecure channel.

Password-based authentication consists of the Public Key Infrastructure (PKI) and Password-only models as presented in Figure 2.1 below. In PKI, the client shares a password with the server and has the server's public key. The public key of the server is used by the client to encrypt the password, then sends it across to the server. The first researchers to present this model are Gong et al. [10], [11]. There protocol concentrated on resisting offline dictionary attacks but lacks security proofs for the

model. Halevi and Krawczyk [12] filled this gap, and they became the number one to present thorough proof of security for the setting.

Bellovin et al. [13] in 1992 proposed the second model. In this model, authentication is based on password-only, and it uses the password to encrypt randomly generated numbers for the goal of key exchange protocol. Bellovin and Merritt [13] model lacked security model and Bellare et al. [14] and Boyko et al. [15] filled this gap. These password-only authenticated protocols were not both practical and secure. Katz et al. [16] in 2001 came up with one that is practical and secure. These protocols assume that a single server stores all the passwords for authentication. For this reason, all the passwords are exposed when an adversary compromises the server. Yi et al. [17], [18], [19] came up with identity-based setting relating with the identity-based encryption scheme [20] and [21]. In their model, the client has the knowledge of the only the password and the server has the knowledge of both the password and the private key relating of its identity. The client encrypts the password with the server's identity. This setting is a hybrid of the PKI and password-only model.

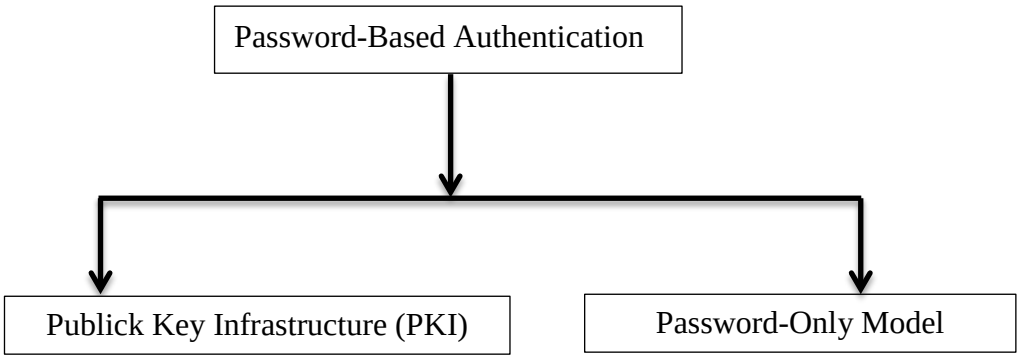

Figure 2.1: Chart for the models of Password-Based Authentication

2.1.2 Two-Server Authenticated Key Exchange

To tackle the issue with single server storing the clients' passwords, Ford and Kaliski [22] in 2000 came up with Password Authenticated Key Exchange (PAKE) protocol based on the public key infrastructure model, the first where n-server jointly authenticate a client. They claimed that their protocol remains secure on the assumption that $n-1$ servers remains compromised out of the n-server, but no formal security proof shown for the protocol. Then, MacKenzie et al. [23] in 2002 proposed a protocol based on the PKI model where only t out of n-server collaborated in the authentication process. This protocol is secure given that adversary attacks only $t-1$ servers and they provided a formal proof for their protocol within the random oracle model.

Based on the PKI model, Brainard et al. [1] in 2003 fostered the first two-server protocol. Their protocol uses the assumption that there exists a secure channel between the communicating client and servers. Based on password-only model, Katz et al. [4] in 2012 proposed two-server password-only authenticated key exchange protocol. Their protocol was the first two-server protocol in password-only model with security proofs, built on the single-server password-based key-exchange protocol of Katz-Ostrovsky-Yung [16] known as the KOY protocol. The scheme of Katz et al. [4] shares client's randomly selected password, pwc amongst the two servers $S_A \; and \; S_B$. These two servers collaborate to verify the identity of the client since it shares its password between them and they finally arrive at a joint key. The scheme of [4] is symmetric but lacks efficiency in practical use.

In 2013, Yi et al. [5] proposed a new symmetric two-server password-only authenticated key exchange protocol that enables two-server architecture compute in parallel. Their protocol claims to be efficient in practical use than the existing Katz et al.'s protocol [4] because of its parallelism in computation. We give a detailed review of [5] protocol in Section 2.2 in an effort to investigate their method of authenticated key exchange protocol.

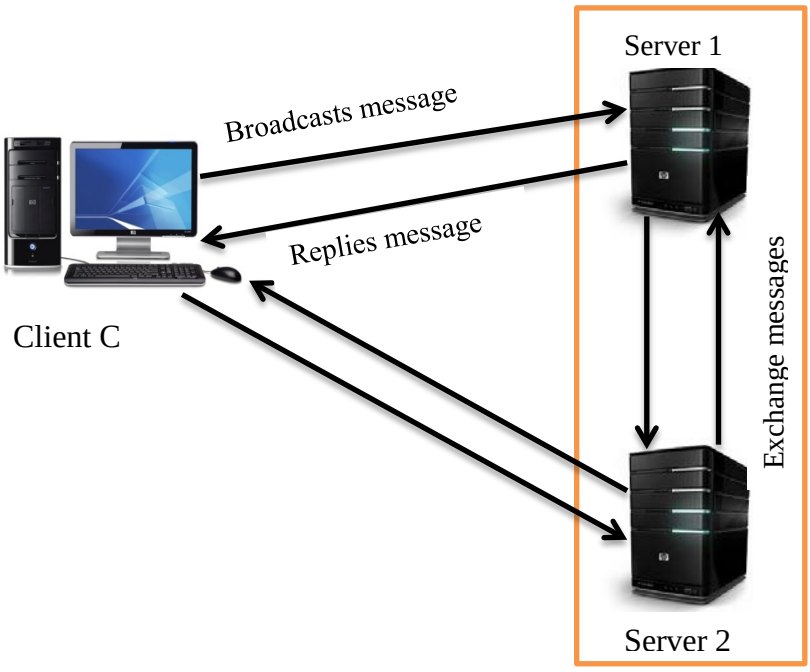

Figure 2.2: Two-server key exchange protocol

In Figure 2.2, the client, C that is located anywhere on the network before authentication, chooses a password and computes authenticators for server 1 and server 2 in such a way that the password will not revealed to anyone except server 1 and server 2 conspire. It sends the authenticator across to the servers through a secure a channel during registration. The two servers, which may be on the same network with the client, conjointly authenticate the client during authentication phase based

on the authentication information supplied by the client during registration. During authentication, the client broadcasts messages to the two servers. The two servers then based on the message from the client, perform some computations and exchange messages in parallel for authenticating the client. This parallelism in message exchange reduces computation time and increases performance. At the end of computations, the two servers and the client arrive at a shared secret key used in securing their subsequent communications. If an adversary compromises any of the servers, it can never fool the other server for being the registered client and this is one of the main advantages of two-server over traditional single server for authentication and key exchange.

2.1.3 Diffie Hellman Key Exchange Scheme

Whitfield Diffie et al. conceived the key exchange protocol [7] in 1976, a simple public key algorithm. Their key exchange scheme is the foremost that is practical for the purpose of enabling communicating parties (two users) to establish a secret key for subsequent communication over an insecure channel, making use of a public key scheme on discrete logarithms. The effectiveness of this protocol depends on the difficulty of computing discrete logarithm. This protocol is secure when the two communicating parties remain authenticated. It is non-authenticated key exchange protocol but provided the basis for most authenticated key exchange protocols.

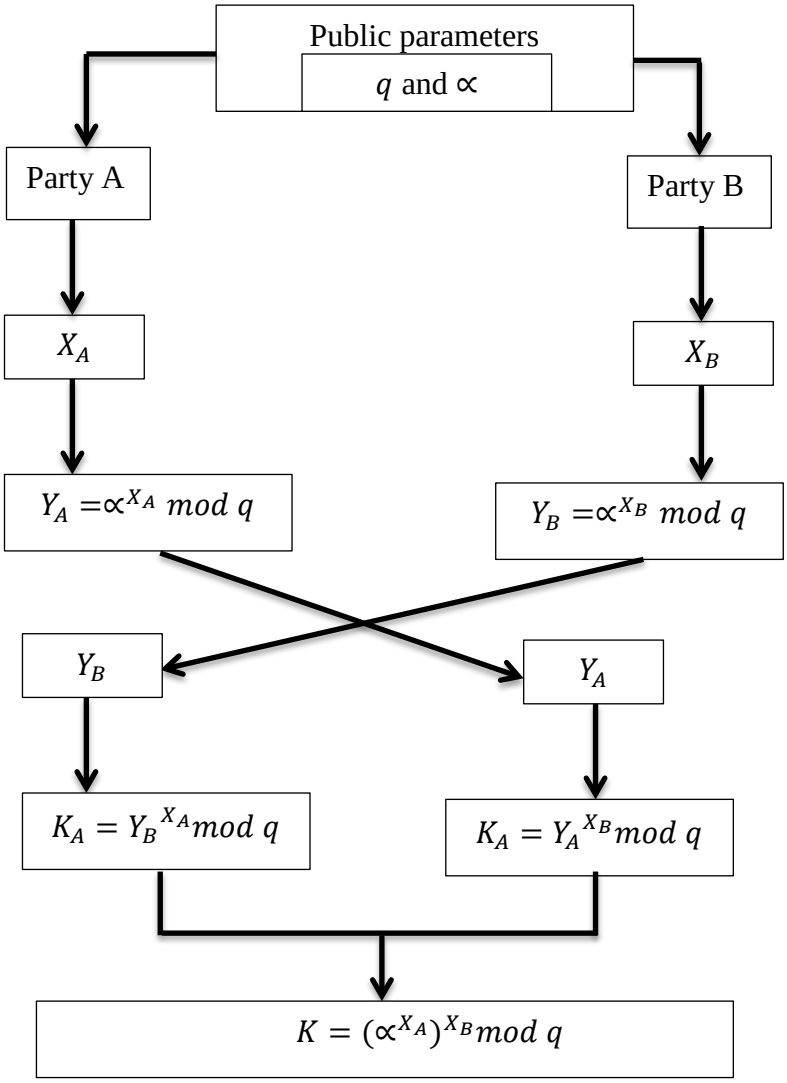

Figure 2.3: Diagram illustrating Diffie Hellman Key Exchange

The algorithm is such that communicating party A and party B agree on a primitive root, $\propto$ of the prime q, where $\propto < q$. Primitive root $\propto$ is a generator of the cyclic group of prime q. Party A selects X_A in a way that $X_A < q$, then computes $Y_A = \propto^{X_A} mod\ q$, where X_A and Y_A are the private and public keys respectively. In the same way, Party B selects X_B with $X_{\boldsymbol{B}} < q$, then computes $Y_{\boldsymbol{B}} = \propto^{X_B} mod\ q$. Party A and party B exchange Y_A and Y_B. After receiving Y_B, party A calculates the secret key $K_A = {Y_B}^{X_A} mod\ q$ and party B, after receiving Y_A calculates the secret key $K_B = {Y_A}^{X_B} mod\ q$. It can be seen that $K_A = K_B$ since Y_A and Y_B are exchange by both

parties and each party knows its private key. Then party A and party B agree on a common secret key to protect their subsequent communications. We can summarize this common secret key with the relation $K = (\propto^{X_A})^{X_B} mod\ q$, since $(\propto^{X_A})^{X_B} mod\ q = (\propto^{X_B})^{X_A} mod\ q$. We illustrated the algorithm with Figure 2.3 above. For party A to determine the private key, X_B of party B, calculation of discrete logarithm is involved, which is a difficult problem.

2.1.4 ElGamal Encryption Scheme

The encryption scheme proposed by ElGamal in 1985 [6] is based on the key exchange scheme of Diffie-Hellman [7]. He presented a system that rests on the complexity of calculating discrete logarithms on finite fields just like Diffie-Hellman. ElGamal scheme [6] consists of three stages, the key generation, encryption and decryption. Prior to key generation, party A and B jointly generate public parameters q and $\propto$ much like Diffie-Hellman. The primitive root, α of the prime, q is the generator of the cyclic group, G based on q. During key generation, party A selects the key, X_A for decryption in a way that $X_A < q$, then it computes the key, $Y_A = \propto^{X_A} mod\ q$ for encryption. The private key of party A is X_A and the public key is Y_A. To encrypt a message M during encryption stage, party A chooses an integer $r < q$, computes a key $K = (Y_A)^r mod\ q$, and using the encryption key, Y_A performs the encryption $C = E(M, Y_A) = (A, B)$, where A and B are computed as $A = \propto^r mod\ q$ and $B = (K \cdot M) mod\ q$ respectively. During decryption, party B will first computing $K = A^{X_A} mod\ q$ and reverse the encryption process to obtain the resulting plaintext $M = D(C, X_A) = (B/K) mod\ q$, which is equal to $(B \cdot K^{-1}) mod\ q$. We recall that $B = (K \cdot M) mod\ q$, therefore, $M = (K \cdot M \cdot K^{-1}) mod\ q = M$.

2.2 Review of YLW Protocol

The YLW protocol [5] consists of three major stages – the stage of initializing all processes, the stage for registration, and the stage for authentication and exchange of keys. In each phase, the communicating parties perform some computations leading ultimately to establishing shared secret keys. Computations are not explicitly specifying modulo q in the protocol, but assumed. We reviewed each of these phases in the following sections to enable understanding the problem with the protocol.

2.2.1 Phase 1: Initialization

This phase is about the sequence of actions from the two servers, during which they generate and publish public system parameters. So, the two servers, S_i ($i = 1, 2$) mutually use the generator, g_1 to generate a cyclic group, G based on a large prime number, q as well as a hash function H:$\{0, 1\}^* \rightarrow \mathbb{Z}_q$.

After choosing the cyclic group, S_i (i=1, 2) choose an integer s_i (i=1, 2) randomly from $\mathbb{Z}_q^*$. The servers, S_1 computes ${g_1}^{s_1}$mod q and S_2 computes ${g_1}^{s_2}$ *mod* q and exchange the resulting values. The servers make public the following parameters such that the client has access to them: G, q, g_1, g_2, and H, where g_2 is computed using

$$g_2 = {g_1}^{s_1 s_2} \ mod \ q. \tag{2.1}$$

The servers, S_i (i = 1, 2) now have a key, left hand of (2.1) to enable communication with each other.

2.2.2 Phase 2: Registration

The client, C is the only one involved in this phase, during which it registers at both servers, S_i (i = *1, 2*) through a secure channel. Decryption and encryption keys, (x_i, y_i) are generated by the client, C for the servers, S_i (i = *1, 2*), where the encryption keys, y_i (i=1, 2) are computed using equation below:

$$y_i = g_1{}^{x_i} mod\ q. \tag{2.2}$$

It selects a password, *pwc*, then encodes it according to ElGamal encryption scheme (see (1), (2) in [6]) using the keys y_i (i=1, 2) with a_i (i =1, 2) selected at random from $\mathbb{Z}_q^*$. The encryption is performed using equations (2.3) and (2.4) below:

$$A_i = g_1^{a_i}\ mod\ q, \tag{2.3}$$

$$B_i = g_2^{pwc} y_i^{a_i}\ mod\ q. \tag{2.4}$$

The Client then selects arbitrarily b_1 from $\mathbb{Z}_q^*$, computes

$$b_2 = H(pwc) \oplus b_1, \tag{2.5}$$

moreover, sends authenticators to the two servers as represented in the equations (2.6) and (2.7):

$$C \rightarrow S_1 : Auth_C^{(1)} = \{x_1, a_1, b_1, (A_2, B_2)\} \tag{2.6}$$

$$C \rightarrow S_2 : Auth_C^{(2)} = \{x_2, a_2, b_2, (A_1, B_1)\}. \tag{2.7}$$

After the client making available the authentication information to the servers, the two servers are ready to authenticate the client and establish shared secret keys.

2.2.3 Phase 3: Authentication and Key Exchange

The authentication and key exchange phase in [5] involves five steps of sequence of actions from both the servers and the client. These steps are as follows:

Step 1: the client, C has to choose r randomly from $\mathbb{Z}_q^*$, computes

$$R = g_1^r g_2^{-pwc} mod\ q, \tag{2.8}$$

and broadcasts a request message, M_1 to S_i ($i = 1, 2$) as presented in equation (2.9) below:

$$C \rightarrow S_i: M_1 = \{C, Req, R\}. \tag{2.9}$$

Step 2: The server, S_1 chooses r_1 at random from $\mathbb{Z}_q^*$, computes

$$A_2' = A_2^{r_1} mod\ q, \tag{2.10}$$

$$B_2' = (R \cdot B_2)^{r_1} mod\ q, \tag{2.11}$$

then prepares the message below based on the resulting encryption values in (2.10) and (2.11):

$$M_2 = \{ A_2',\ B_2' \}. \tag{2.12}$$

Also, the server, S_2 chooses r_2 at random from $\mathbb{Z}_q^*$, computes

$$A_1' = A_1^{r_2} mod\ q, \tag{2.13}$$

$$B_1' = (R \cdot B_1)^{r_2} mod\ q, \tag{2.14}$$

then prepares the message below based on the encryption results from (2.13) and (2.14):

$$M_3 = \{ A_1',\ B_1' \}. \tag{2.15}$$

The both servers, S_1 and S_2 exchange messages (2.12) and (2.15) for further computations.

Step 3: The servers, S_i (i = 1, 2) choose r_i' at random from $\mathbb{Z}_q^*$, compute

$$R_i = A_i'^{a_i^{-1} r_i'} mod\ q, \tag{2.16}$$

$$K_i = (B_i'/A_i'^{x_i})^{r_i'} mod\ q, \tag{2.17}$$

$$h_i = H(K_i, 0) \oplus b_i, \tag{2.18}$$

then reply the message $M_{3+\mathrm{i}}$ to the client, *C* for i = 1, 2

$$S_i \rightarrow C\text{: } M_{3+\mathrm{i}} = \{S_i, R_i, h_i\}. \tag{2.19}$$

Step 4: The client, C computes the following for i = 1, 2 after receiving messages (2.19) from the two servers:

$$K_i' = R_i^r mod\ q, \tag{2.20}$$

and, checks if the left hand of (2.21) is equal to the right hand,

$$H(K_1', 0) \oplus H(K_2', 0) \oplus h_1 \oplus h_2 = H(pwc). \tag{2.21}$$

The servers, S_i (i =*1, 2*) are considered to be authentic if equality (2.21) holds. Then the client, *C* computes:

$$h_i' = H(K_i', 1) \oplus H(K_i', 0) \oplus h_i, \tag{2.22}$$

broadcasts the message in (2.23) to the two servers, S_i (i = 1, 2)

$$\mathrm{C} \rightarrow S_i\text{: } M_6 = \{ h_1',\ h_2' \}, \tag{2.23}$$

establishes secret session keys with the servers, S_i (i =*1, 2*) as in equation (2.24):

$$SK_i' = H(K_i', 2). \tag{2.24}$$

Step 5: the two servers, S_i (i = 1, 2) check if equality (2.25) below holds after receiving the message in (2.23) and conclude the authenticity of the client, C, otherwise the client is not authentic:

$$H(K_i, 1) \oplus b_i = h_i'. \tag{2.25}$$

Finally, the servers, S_i (i = *1, 2)* and the client, C agrees together on confidential session keys in (2.26) below;

$$SK_i = H(K_i, 2). \tag{2.26}$$

Left-hand side of (2.24) is equal to the left-hand side of (2.26) because left-hand side of (2.17) is equal to the left-hand side of (2.20) (see Section 4.2.4, p. 1779, left column in [5]).

Chapter 3

PROBLEM WITH THE REVIEWED PROTOCOL

In Chapter 2, see (2.3) and (2.4) in Section 2.2.2, the client, *C* chooses a_i (*i=1*, *2*) randomly from $\mathbb{Z}_q^*$ without restriction, and used them to encrypt the chosen password, pwc "according to ElGamal encryption" [5, p. 1777, Section 4.2.2]. In the proof of Theorem 1[5, p. 1778, right column], it is shown that $K_1 = K_1'$ (see (2.17) and (2.20)) since, from (2.2) - (2.4), (2.8), (2.13), and (2.14),

$$K_1 = g_1^{rr_1'r_2} \ mod\ q, \tag{3.1}$$

from (2.3), (2.13), and (2.16),

$$K_1' = R_1^r = (g_1^{r_1'r_2a_1a_1^{-1}})^r = g_1^{rr_1'r_2a_1a_1^{-1}} = g_1^{rr_1'r_2} \ mod\ q. \tag{3.2}$$

However, the last equality in (3.2) is valid only when

$$rr_1'r_2a_1a_1^{-1} = rr_1'r_2 \text{ mod } (q\text{-}1) \tag{3.3}$$

(see, e.g., (5), (6) in [6]) since the inverse of a_1 is used in the exponent of equation (3.2). As far as a_1 is selected and used according to (2.3) and (2.4) from $\mathbb{Z}_q^*$, its inverse modulo *q* exists and is used in (2.16), (3.2), and (3.3).

It is known in [24] that the inverse of an integer depends on modulo with respect to which it is considered. In (2.16), it is not specified modulo with respect to which inverses of $a_i(i = 1,2)$ are calculated. In the description of the YLW protocol [5], modulo operations are not shown explicitly, but assumed as modulo *q*. Hence, inverses of $a_i(i = 1,2)$ is also to be modulo *q*. Actually, all numbers below q are

invertible modulo q and hence can be selected randomly as it is supposed since they are invertible modulo q (see Section 2.2.2, Registration phase). In that case, the left hand side of (3.3) is,

$$rr_1'r_2a_1a_1^{-1} = rr_1'r_2(1 + nq) = rr_1'r_2 + nqrr_1'r_2 \quad (3.4)$$

for some integer *n*, and may not be equal to the right hand side of (3.3) modulo Euler's totient function $\varphi(q) = q - 1$, for which [24] and any *a, k*

$$a^{k\varphi(q)} mod\ q = 1 \quad (3.5)$$

holds.

The source of the problem with YLW protocol [5] is that parameters used in its exponents are not considered modulo Euler's totient function $\varphi(q) = q - 1$. In the following Sections 3.1, 3.2, and 3.3, we present a numerical counter-example that actually illustrates the failure of YLW protocol due to using congruency of the powers in (3.2) modulo *q* instead of (*q-1*).

3.1 Sequence of Activities from the Servers

In this Section, we present numerical counter-example for the initialization phase of the protocol as described in Section 2.2.1 of Chapter 2. The two servers perform all activities during this stage.

The servers, S_1 and S_2 decide on a cyclic group, G = {1, 2,…, 12} generated by the generator $g_1 = 2$, based on large prime q = 13 (see Chapter 2, Section 2.2.1). Server, S_1 chooses s_1 = 2 and server, S_2 chooses s_2 = 3 randomly from $\mathbb{Z}_q^*$ and exchange messages $S_1 \rightarrow S_2$: ${g_1}^{s_1}$ and $S_2 \rightarrow S_1$: ${g_1}^{s_2}$ to arrive at g_2 = 12. The computation is according to (2.1): $g_2 = {g_1}^{s_1 s_2} mod 13 = 2^{2\cdot 3} mod\ 13 = 64 mod 13 = 12$. The

servers, S_1 and S_2 together publish public system parameters, which will be accessible to both communicating parties as follows:

G= {1, 2,…,12}, q = 13, g_1 = 2, g_2 = 12, H:{0, 1}* → $\mathbb{Z}_q$.

3.2 Sequence of Activities from the Client

The client, C starts by generating decryption and encryption keys x_1 = 2, x_2 = 3, and

$y_1 = {g_1}^{x_1} mod 13 = 2^2 mod 13 = 4$,

$y_2 = {g_1}^{x_2} mod 13 = 2^3 mod 13 = 8$ respectively (see equation (2.2)). We choose the password, pwc = 3, a_1= 6, and a_2= 6 randomly from $\mathbb{Z}_q^*$ for the client and encrypt the password to obtain:

$A_1 = g_1^{a_1} mod 13 = 2^6 mod 13 = 12$,

$B_1 = g_2^{pwc} y_1^{a_1} mod 13 = 12^3 \cdot 4^6 mod 13 = 12$,

$A_2 = g_1^{a_2} mod 13 = 2^6 mod 13 = 12$,

and $B_2 = g_2^{pwc} y_2^{a_2} mod 13 = 12^3 \cdot 8^6 mod 13 = 1$

(see Chapter 2, Section 2.2.2, equations (2.3) and (2.4)).

The client, C chooses $b_1 = 5$ at random from $\mathbb{Z}_q^*$ and computes b_2 = H(pwc) $\oplus$ b_1 corresponding to equation (2.5). At the end of these computations, the client, C delivers authenticators, $Auth_C^{(1)}$ = $\{2, 6, 5, (12, 1)\}$ to S_1 and $Auth_C^{(2)}$ = $\{3, 6, b_2, (12, 12)\}$ to S_2 agreeing with equations (2.6) and (2.7). The sending of the above authenticators to the servers by the client marks the end of the registration phase in the protocol [5].

3.3 Sequence of Activities from the Servers/Client

In this section, we illustrate the actions between the client and the servers on the authentication and key exchange phase. For the client, C: we choose r = 5 randomly from $\mathbb{Z}_q^*$ and compute *R* as follows using equation (2.8):

$R = g_1^r g_2^{-pwc} mod\ q = g_1^r g_2^{-1(pwc)} mod\ q = 2^5 \cdot 12^3 mod 13 = 7.$

It then relays the message, M_1 = {C, Req, 7} to the two servers (see Chapter 2, Section 2.2.3, step 1).

For server, S_1: we choose $r_1 = 3$ randomly from $\mathbb{Z}_q^*$, compute A_2' and B_2' following equations (2.10) and (2.11) as shown below:

$A_2' = A_2^{r_1} mod q = 12^3 mod 13 = 12,$

$B_2' = (R \cdot B_2)^{r_1} mod 13 = (7 \cdot 1)^3 mod 13 = 7^3 mod 13 = 5$

and S_1 prepares message $M_2 = \{12, 5\}$ as in equation (2.12).

For server, S_2: we choose $r_2 = 7$ randomly from $\mathbb{Z}_q^*$, compute A_1' and B_1' according to equations (2.13) and (2.14) as shown below:

$A_1' = A_1^{r_2} mod q = 12^7 mod 13 = 12,$

$B_1' = (R \cdot B_1)^{r_2} mod 13 = (7 \cdot 12)^7 mod 13 = (84)^7 mod 13 = 7,$

and S_2 prepares message $M_3 = \{12, 7\}$ according to equation (2.15).

Server, S_1 and S_2 exchange messages M_2 and M_3 respectively (see Chapter 2, Section 2.2.3 step 2).

Now that S_1 has message, $M_3 = \{12, 7\}$, it chooses $r_1' = 3$ randomly from $\mathbb{Z}_q^*$, and computes R_1, K_1 and h_1 according to equations (2.16), (2.17), and (2.18):

$R_1 = A_1'^{a_1^{-1} r_1'} mod 13 = 12^{11 \cdot 3} mod 13 = 12^{33} mod 13 = 12,$

$K_1 = (B_1' / A_1'^{x_1})^{r_1'} mod q = (B_1' \cdot A_1'^{-1(x_1)})^{r_1'} = (7 \cdot 12^2)^3 mod 13 =$

$(1008)^3 mod 13 = 5,$

$h_1 = H(K_1, 0) \oplus b_1,$

it then replies the message, $M_4 = \{S_1, 12, h_1\}$ according to (2.19) to the client.

Also, server S_1 having the message, $M_2 = \{12, 5\}$ chooses r_2' = 6 randomly from $\mathbb{Z}_q^*$ and computes R_2, K_2 and h_2 according to equations (2.16), (2.17), and (2.18):

$R_2 = A_2'^{a_2^{-1} r_2'} mod 13 = 12^{11 \cdot 6} mod 13 = 12^{66} mod 13 = 1,$

$K_2 = (B_2' / A_2'^{x_2})^{r_2'} mod q = (B_2' \cdot A_2'^{-1(x_2)})^{r_2'} = (5 \cdot 12^3)^6 mod 13 = 12,$

$h_2 = H(K_2, 0) \oplus b_2,$

moreover, replies the message, $M_5 = \{S_2, 1, h_2\}$ according to (2.19) to the client, C. (see Chapter 2, Section 2.2.3, step 3).

The client, with the messages M_4 and M_5 from S_1 and S_2 respectively available, computes K_1' and K_2' according to (2.20):

$K_1' = R_1^r mod q = 12^5 mod 13 = 12$

$K_2' = R_2^r mod q = 1^5 mod 13 = 1$ (see Chapter 2, Section 2.2.3, step 4).

3.4 Observations

From the computations so far in the last three Sections (3.1, 3.2, and 3.3), it could be observed that the value of K_1' is not equal to the value of K_1; value of K_2' is not equal to value of K_2 (see Section 3.3), which are meant to be equal to enable parties arrive at same secret keys in (2.24) and (2.26) at the end of computations. The difference in values of K_1', K_2', K_1, and K_2 is as a result of computations involving a_i (i = *1, 2*) based on exponents congruent modulo *q*. This shows that choosing a_i (i = *1, 2*) at random from $\mathbb{Z}_q^*$ with its multiplicative inverses involved in computations, may lead parties arriving at different secret keys.

The proof of Theorem 1 [5, p.1778, Section 4.2.4], shows that the right-hand sides of (2.17) and (2.20) are equal, and therefore the secret keys in (2.24) and (2.26) are same. From (2.3), (2.4), (2.8), (2.13), and (2.14) respectively for S_1, we have

$$A_1' = (g_1^{a_1})^{r_2} = g_1^{r_2 a_1} \ (mod\ q), \tag{3.6}$$

$$B_1' = (g_1^{r} g_2^{-pwc} g_2^{pwc} y_1^{a_1})^{r_2} = g_1^{r r_2} y_1^{r_2 a_1} \ (mod\ q). \tag{3.7}$$

From (3.6) and (2.16), we have

$$R_1 = (g_1^{r_2 a_1})^{a_1^{-1} r_1'} = {g_1}^{r_1' r_2}, \tag{3.8}$$

where $a_i a_i^{-1}$ varnishes.

Taking $R_1 = (g_1^{r_2 a_1})^{a_1^{-1} r_1'}$ from (3.8) and using the values $q = 13$, $g_1 = 2, a_1 = 6, r_2 = 7, r_1' = 3$ defined in the Sections 3.1, 3.2, and 3.3, we have $R_1 = (2^{7 \cdot 6})^{6^{-1} mod 13 \cdot 3}\ mod\ 13 = (2^{7 \cdot 6})^{11 \cdot 3}\ mod\ 13$

$= (2^{4 \cdot 10} \cdot 4)^{33}\ mod\ 13 = (3^{10} \cdot 4)^{33}\ mod\ 13 = (3^3 \cdot 3 \cdot 4)^{33}\ mod\ 13 =$

$12^{33}\ mod\ 13 = 12$, which is same in Section 3.3. But using the right hand side of

(3.8), $R_1 = {g_1}^{r_1' r_2} = 2^{3 \cdot 7}\, mod\ 13 = 2^{4 \cdot 5} \cdot 2\, mod\ 13 = 3^5 \cdot 2\, mod\ 13 = 9 \cdot 2\, mod\ 13 = 5$, which is not equal to 12, previously obtained. Thus, (3.8) allegedly proved in Section 4.2.4 of [5] is not true, and $R_1 = A_1'^{a_1^{-1} r_1'} = (g_1^{r_2 a_1})^{a_1^{-1} r_1'} \neq {g_1}^{r_1' r_2}$.

The failure of the proof is due to the use of multiplicative inverse of the exponent a_1 modulo q instead of using multiplicative inverse modulo Euler's totient function $\varphi(x)$, which defines the number of numbers less than *x* and relatively prime to *x*, which is for the case under consideration, $\varphi(q) = q - 1$. If we use multiplicative inverse modulo *q-1*, we get $R_1 = (g_1^{r_2 a_1})^{a_1^{-1} r_1'}$

$= (2^{7 \cdot 6})^{6^{-1} mod 12 \cdot 3} mod\ 12$, which is not defined since $6^{-1}\ mod\ 12$ does not exist. Hence, just finding inverses modulo (*q-1*) is not sufficient to fix the protocol. It's fixing is proposed in next chapter.

Chapter 4

MODIFICATION OF THE PROTOCOL

In this Chapter, due to the failure of the protocol [5] as presented in Chapter 3, we proposed a modification and provided a numerical example to illustrate the correctness and efficiency of the proposed modification using the settings in previous Chapter.

4.1 Proposed Modification

We propose that a_i (i = 1, 2) should be chosen from $\mathbb{Z}_q^*$ such that the condition of relative primality

$$\gcd(a_i, q-1) = 1 \tag{4.1}$$

holds.

Hence, in Chapter 2, Section 2.2.2, instead of writing after (2.4) "with a_i (i =1, 2) randomly chosen from $\mathbb{Z}_q^*$", we should write "with relatively prime to (*q-1*) values a_i (i =1, 2) meeting (4.1) and randomly chosen from $\mathbb{Z}_q^*$." We see that our choice of a_i (i =1, 2) in Chapter 3, Section 3.2, violates (4.1), and we have the failure of the protocol. If (4.1) holds, due to (3.5), the proof of (3.8) and of Theorem 1 in Section 4.2.4 of [5] is correct since equality (3.8) was the only one problem in the proof. Also, if (4.1) holds, value of K_1 will be equal to value of K_1', thereby parties arriving to same secret session keys in (2.24) and (2.26). Therefore, choosing $a_i (i = 1,2)$

should be such that $a_i a_i^{-1} \equiv 1\, mod\ (q-1)$ exists for $i = 1,2$. We provided a numerical example in the following section for illustration.

4.2 Numerical Example for the Modification

We provided an example to prove the working of the modified protocol using the same settings in Chapter 3, Sections 3.1, 3.2, and 3.3.

Let G= {1, 2,…,12}, q = 13, g_1 = 2, g_2 = 12 (settings from Section 3.1).

Let x_1 = 2, x_2 = 3, $y_1 = 4$, $y_2 = 8$, $b_1 = 5$, b_2 = H(pwc) $\oplus$ b_1, pwc = 3, but with a_1= 7, and a_2= 7, which are relatively prime to q – 1 = 12 and $a_1 a_1^{-1} \equiv 1\, mod\ (q-1)$ (settings from Section 3.2). The client encrypts the password to obtain:

$A_1 = g_1^{a_1} mod 13 = 2^7 mod 13 = 11$, $B_1 = g_2^{pwc} y_1^{a_1} mod 13 = 12^3 \cdot 4^7 mod 13 = 9$,

$A_2 = g_1^{a_2} mod 13 = 2^7 mod 13 = 11$, and

$B_2 = g_2^{pwc} y_2^{a_2} mod 13 = 12^3 \cdot 8^7 mod 13 = 8$ (see Chapter 2, Section 2.2.2, equations (2.3) and (2.4)).

The client, C delivers authenticators, $Auth_C^{(1)} = \{2, 7, 5, (11, 8)\}$ to S_1 and $Auth_C^{(2)} = \{3, 7, b_2, (11, 9)\}$ to S_2 according to (2.6) and (2.7).

Let r = 5, $r_1 = 3$, $r_2 = 7$, $r_1' = 3$, r_2' = 6 (settings from Section 3.3).

The client computes *R* according to (2.8):

$R = g_1^r g_2^{-pwc} mod\ q = g_1^r g_2^{-1(pwc)} mod\ q = 2^5 \cdot 12^3 mod 13 = 7$, then broadcast message, M_1 = {C, Req, 7} to the two servers.

S_1 computes:

$A'_2 = A_2^{r_1} mod q = 11^3 mod 13 = 5,$

$B'_2 = (R \cdot B_2)^{r_1} mod 13 = (7 \cdot 8)^3 mod 13 = 4^3 mod 13 = 12$ and prepares message $M_2 = \{5, 12\}$ (see (2.10), (2.11), and (2.12)).

S_2 computes:

$A'_1 = A_1^{r_2} mod q = 11^7 mod 13 = 2,$

$B'_1 = (R \cdot B_1)^{r_2} mod 13 = (7 \cdot 9)^7 mod 13 = (11)^7 mod 13 = 2,$ and prepares message $M_3 = \{2, 2\}$ (see (2.13), (2.14), and (2.15)). S_1 and S_2 exchange messages M_2 and M_3 respectively (see Chapter 2, Section 2.2.3 step 2).

S_1 prepares message, $M_3 = \{2, 2\}$ and computes R_1, K_1 and h_1 according to (2.16), (2.17), and (2.18):

$R_1 = A_1'^{a_1^{-1} r'_1} mod 13 = 2^{7^{-1} mod 12 \cdot 3} mod 13 = 2^{7 \cdot 3} mod 13 = 2^{21} mod 13 = 5,$

$K_1 = (B'_1 / A_1'^{x_1})^{r'_1} mod q = (B'_1 \cdot A_1'^{-1(x_1)})^{r'_1} = (2 \cdot 7^2)^3 mod 13 = (98)^3 mod 13 = 5,$

$h_1 = H(K_1, 0) \oplus b_1,$

it then replies the message, $M_4 = \{S_1, 5, h_1\}$ according to (2.19) to the client.

S_2 prepares message, $M_2 = \{5, 12\}$ and computes R_1, K_1 and h_1 according to (2.16), (2.17), and (2.18):

$R_2 = A_2'^{a_2^{-1} r'_2} mod 13 = 5^{7^{-1} mod 12 \cdot 6} mod 13 = 5^{7 \cdot 6} mod 13 = 5^{42} mod 13 = 12,$

$K_2 = (B'_2 / A_2'^{x_2})^{r'_2} mod q = (B'_2 \cdot A_2'^{-1(x_2)})^{r'_2} = (12 \cdot 8^3)^6 mod 13 = 12,$

$h_1 = H(K_1, 0) \oplus b_1,$

moreover, replies the message, $M_5 = \{S_2, 12, h_1\}$ according to (2.19) to the client, C.

The client computes K_1' and K_2' according to (2.20):

$K_1' = R_1^r mod q = 5^5 mod 13 = 5$

$K_2' = R_2^r mod q = 12^5 mod 13 = 12.$

We observed from the above computations that the value of $K_1' = 5$ is equal to the value of $K_1 = 5$; value of $K_2' = 12$ is equal to value of $K_2 = 12$, because the condition $a_i a_i^{-1} \equiv 1\ mod\ (q-1)$, hence the communicating parties will arrive at same secret keys at the end of the computations.

Let us prove (3.8) using (3.5):

$R_1 = (g_1^{r_2 a_1})^{a_1^{-1} mod(q-1) r_1'} = g_1^{r_2 a_1 a_1^{-1} mod(q-1) r_1'} = g_1^{r_2(1+k(q-1))r_1'} = g_1^{r_2 r_1'} g_1^{k(q-1)}$

$= g_1{}^{r_1' r_2}\ mod\ q$. Using the same settings above, let us illustrate (3.8) to show the correctness of the protocol with the modification.

Taking $R_1 = (g_1^{r_2 a_1})^{a_1^{-1} r_1'}$ from (3.8), we have,

$R_1 = (2^{7\cdot 7})^{7^{-1} mod 12\cdot 3}\ mod\ 13 = (2^{7\cdot 7})^{7\cdot 3}\ mod\ 13 = 2^{21}\ mod\ 13 = 5$ that is same as the result above, in this section. Also, using the right hand side of (3.8), $R_1 = g_1{}^{r_1' r_2} = 2^{3\cdot 7}\ mod\ 13 = 2^{4\cdot 5} \cdot 2\ mod\ 13 = 3^5 \cdot 2\ mod\ 13 = 9 \cdot 2\ mod\ 13 = 5$ that is same as the right hand side of (3.8) calculated above. Therefore, equation (3.8) as proved in [5, p.1778, Section 4.2.4] is correct when $a_i (i = 1,2)$ is chosen such that $a_i a_i^{-1} \equiv 1\ mod\ (q-1)$, that is, congruency of exponent modulo Euler's totient function. It could be concluded here that the modification will make the protocol work efficiently.

Chapter 5

CONCLUSION

We have presented the password authenticated key exchange in this research work, with much attention on the efficient two-server password-only authenticated key exchange of Xun Yi, San Ling, and Huaxiong Wang, the most recent two-server authenticated key exchange in the password-based model. We have done a review of the protocol and discovered a problem that will cause the failure of the protocol. This problem was due to the application of inverses in the exponent modulo q, instead of modulo Euler's totient function in the protocol leading to unequal result in computations that will make parties arrive at different secret session keys. The congruency of exponents was not considered modulo Euler's totient function in computations where inverses of the parameter are used. We provided a counter-example to illustrate the problem, and proposed a modification of the protocol. We fixed this problem by placing a restriction on how to choose randomly the parameter, a_i from all the elements that are invertible in the cyclic group, since its multiplicative inverse is used in computations. We provided a proof and numerical example to illustrate that the modification will work without failure. With such proposed modification, the protocol works correctly and is efficient.

5.1 Future Work

We recommend that future research should concentrate on the security proof of the protocol to verify if it is secure against passive and active attack. We also recommend the implementation of the both protocol to verify its practical efficiency and correctness and the performance analysis checked. More research interest should focus on n-server password-only authenticated key exchange protocol and a two-server authenticated key exchange on the PKI model. Finally, we recommend that research should incorporate elliptic curve cryptography in this protocol to see its outcome.

REFERENCES

[1] Brainard, J, A Jueles, B S Kaliski, and M Szydlo, "A New Two-Server Approach for Authentication with Short Secret," *Proc. 12th Conf. USENIX Security Symp,* pp. 201-214, 2003.

[2] Yang, Y, F Bao, and R H Deng, "A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprises," *Proc. 20th IFIP Int'l Information Security Conf. (SEC '05),* pp. 95-111, 2005.

[3] Yang, Y, R H Deng, and F Bao, "A Practical Password-Based Two-Server Authentication and Key Exchange System," *IEEE Trans. Dependable and Secure Computing,* vol. 3, no. 2, pp. 105-114, Apr.-June 2006.

[4] Katz, J, P MacKenzie, G Taban, and V Gligor, "Two-Server Password-Only Authenticated Key Exchange," *Journal of Computer and System Sciences,* vol. 78, pp. 651-669, 2012.

[5] Yi, X, S Ling, and H Wang, "Efficient Two-Server Password-Only Authenticated Key Exchange," *IEEE Transactions on Parallel and Distributed System,* vol. 24, no.9, pp. 1773-1782, 2013.

[6] ElGamal, T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Inf. Theory.* vol. 31, no.4, pp. 469-472, 1985.

[7] Diffie, W, and M E Hellman, "New Directions in Cryptography," *IEEE Trans. on Information Theory,* vol. 22, no. 6, pp. 644-654, November 1976.

[8] McHale, C. (2008), *Secure Communication Concepts Explained Simply*. Retrieved January 14, 2015, from http://ciaranmchale.com/training-courses.html#training-secure-communications

[9] Anderson, R J. (2001), *Security Engineering: A Guide to Building Dependable Distributed Systems,* New York: Wiley Pub.

[10] Gong, L, T M Lomas, R M Needham, and J H Saltzer, "Protecting Poorly-Chosen Secret from Guessing Attacks," *IEEE J. Selected Areas in Comm,* vol. 11, no. 5, pp. 648-656, June 1993.

[11] Lomas, T M, L Gong, J H Saltzer, and R M Needham, "Reducing Risks from Poorly-Chosen Keys," *ACM Operating Systems Rev.,* vol. 23, no. 5, pp. 14-18, 1989.

[12] Halevi, S, and H Krawczyk, "Public-Key Cryptography and Password Protocols," *ACM Trans. Information and System Security,* vol. 2, no. 3, pp. 230-268, 1999.

[13] Bellovin, S, and M Merritt, "Encrypted Key Exchange: Password-Based Protocol Secure against Dictionary Attack," *Proc. IEEE Symp. Research in Security and Privacy,* pp. 72-84, 1992.

[14] Bellare, M, D Pointcheval, and P Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks," *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00)*, pp. 139-155, 2000.

[15] Boyko, V, P Mackenzie, and S Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman," *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00)*, pp. 156-171, 2000.

[16] Katz, J, R Ostrovsky, and M Yung, "Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords," *Proc. Int'l Conf. Theory and Application of Cryptographic Techniques: Advances in Cryptology (Eurocrypt '01)*, pp. 457-494, 2001.

[17] X. Yi, R. Tso, and E. Okamoto, "ID-Based Group Password-Authenticated Key Exchange," *Proc. Fourth Int'l Workshop Security: Advances in Information and Computer Security (IWSEC '09)*, pp. 192-211, 2009.

[18] X. Yi, R. Tso, and E. Okamoto, "Three-Party Password-Authenticated Key Exchange without Random Oracles," *Proc. Int'l Conf. Security and Cryptography (SECRYPT '11)*, pp. 15-24, 2011.

[19] X. Yi, R. Tso, and E. Okamoto, "Identity-Based Password-Authenticated Key Exchange for Client/Server Model," *Proc. Int'l Conf. Security and Cryptography (SECRYPT '12)*, pp. 45-54, 2012.

[20] D. Boneh and M. Franklin, “Identity Based Encryption from the Weil Pairing,” *Proc. 21st Ann. Int’l Cryptology Conf. (Crypto ’01),* pp. 213-229, 2001.

[21] D. Boneh and M. Franklin, “Identity Based Encryption from the Weil Pairing,” *SIAM J. Computing,* vol. 32, no. 3, pp. 586-615, 2003.

[22] Ford, W, and B S Kaliski, "Server-Assisted Generation of a Strong Secret from a Password," *Proc. IEEE Ninth Int’l Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises,* pp. 176-180, 2000.

[23] P. Mackenize, T. Shrimpton, and M. Jakobsson, “Threshold Password-Authenticated key Exchange,” *Proc. 22nd Ann. Int’l Cryptology Conf. (Crypto ’02),* pp. 385-400, 2002.

[24] W. Stallings (2006), *Cryptography and Network Security: Principles and Practice (5th ed.),* pp. 243-251, Boston: Prentice Hall.