Analysis, Design and Implementation of a Voting System Using a Novel Oblivious and Proxy Signature

Olalekan Ihinkalu Ebenezer

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
June 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all 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. Huseyin Öztoprak

3. Assoc. Prof. Dr. Önsen Toygar

ABSTRACT

Electronic Voting System (EVS) makes voting process convenient and more secure. In this thesis, we analyzed and implemented an existing EVS. We used the proxy and oblivious signature for our security. The proxy signature helps curb the aspect of impersonation in the EVS which is a part of the signature that allows A as an original signer to assign her/his signing privilege to someone else called B as a proxy signer. This is very useful since that a scheme will allow an assigned person B (proxy signer) to produce proxy signatures on behalf of the original signer A. Additionally, B (proxy signer) can check the identity of a person R (voter). If the person is eligible to vote, he is given the privilege to exercise her/his franchise, otherwise he or she is denied access from voting. This enables curbing impersonation during the electoral process.

The oblivious signature in the EVS scheme is to help R's (the voter's) choice not to be known by anyone including the proxy signature. This has to do with having n messages as a signature in which R (the voter) could choose 1 -of- n messages to get his message signed while the proxy signer will not be able to find out on which message the voter R has got the signature. The oblivious and proxy signature is efficient in communication, computation and security.

We studied and provided proofs for the Electronic Voting System, made design, implemented and tested the EVS. We also conducted experiments with the EVS. We carried out the experiments based on six phases of the existing EVS time in, compared the existing and the implemented systems. We conclude that the

implemented system has a better computation time (in milliseconds) than that of the existing system.

Keywords: Electronic Voting System (EVS); oblivious and proxy signature; security and privacy.

ÖZ

Elektronik oylama sistemi (EVS) oylama sürecini uygun ve güvenli hale getirir. Bu tezde, mevcut bir elektronik oylama sistemi analiz edilip hayata geçirilmiştir. Güvenlik için vekil ve habersiz imza kullanılmıştır. Vekil (proxy) imza, EVS'de başkasının kimliğine bürünme yönünü engellemeye yardımcı olur. Bu imza, A olarak tanımlanan asıl imzalayıcının imza yetkisini, B olarak tanımlanan ve vekil imzalayıcı olan başka bir kişiye vermesini sağlar. Bu, EVS sistemi için çok faydalıdır, çünkü vekil imzalayıcısı olarak atanan B kişisinin, asıl imzalayıcı A kişisi adına vekil imzalar üretmesini sağlayacaktır. Ek olarak, B olarak tanımlanan vekil imzalayıcı, R olarak tanımlanan seçmenin geçerliği kontrol edebilecektir. Bir kişinin oy kullanma hakkı varsa, kendisine yetkilerini kullanma ayrıcalığı verilir; Aksi takdirde kendisine oy kullanma hakkı verilmez. Bu, seçim sürecinde başkasının kimliğine bürünülmesinin engellenmesini sağlar.

EVS sistemindeki kayıtsız imza, R olarak tanımlanan seçmenlerin seçiminin vekil imzası da dahil hiç kimse tarafından bilinmemesine yardımcı olmaktadır. R seçmeni, n tane mesajdan sadece 1 tanesini seçip imzalayacak ve vekil imzalayıcı hangisini seçip imzaladığını bilemeyecektir. Bu kayıtsız ve vekil imzası, tüm katılımcılar arasında iletişim, hesaplama ve güvenlik açısından etkilidir.

Bu tezde Elektronik Oylama Sistemi üzerinde çalışılmış ve kanıtlar sunulmuştur. EVS sistemi tasarlanmış, uygulanmış ve test edilmştir. Ayrıca EVS sistemi üzerinde deneyler yapılmıştır. Mevcut EVS sisteminin altı aşamasına dayanarak deneyler, yapılmıştır. Mevcut ve uygulanan sonuçlar karşılaştırılmıştır. Elde edilen sonuçların

sistemden daha iyi bir hesaplama süresine (milisaniyede varolan) sahip olduğu sonucuna varılmış.

Anahtar Kelimeler: Elektronik Oylama Sistemi (EVS); kayıtsız ve vekil imza; güvenlik ve mahremiyet

DEDICATED TO GOD AND MY FAMILY FOR THEIR LOVE AND SUPPORT.

ACKNOWLEDGEMENT

First, I want to thank God for His loving kindness, tender mercies and faithfulness for guiding me through this great moment of life.

I will love to extend my gratitude to my supervisor Assoc. Prof. Alexander Chefranov for his fatherly and warm way of supervising me and giving me adequate knowledge to guide me through my thesis.

I would love to thank my parents Mrs. Alice Ihinkalu and my late father Sunday Ihinkalu, my lovely wife, Feyi Ihinkalu and baby boy (Worship); for understanding with me through my program and especially this thesis, I love you all, my siblings, for being there for me. I want to thank Federal University Lokoja, Nigeria in general for making this an historic reality in my life time; I don't take this privilege likely, I also want to thank FUL Computer Science department for their encouragement to undertake this task. I want to appreciate my senior friend John Olaifa and his family for been there for me all through my program, to all my friends Joshua, Bayo, Salwa, Flavien, Jesse Chinda, Psalm and my colleagues in EMU Computer Engineering Department for their contributions to this work. I can't leave out my mentors: Pst. Ola Solomon, Prof. Eric Adewumi, Pst. A. Owolabi. I will love to thank my pastors, Pst. Abiola Abiodun, Pst. Andrew Alola, The music department, Word-Miners unit and family of RCCG Shekinah Parish family in Famagusta. The Joshua's family, Emmanuel's family, John Olododo, Emeka Onwuka, MAV TEAM will not be left out, you guys are the best, finally on this note I also thank everyone who have supported in one way or the other, God bless you greatly, I appreciate you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
BB	Bulletin Board
DRE	Direct Recording Electronics
ElVS	Electrical Voting System
EVS	Electronic Voting System
IV	Internet Voting
RSA	Rivest–Shamir–Adleman algorithm
VS	Voting System

Chapter 1

INTRODUCTION

1.1 General Overview

Today, computer technology has been of tremendous help to humanity. Technology has made electronic voting system much more interesting since it offers more transparency, participation, reduced cost, and quick delivery of results [1].

Almost everything in the world is built on technology; from radio program to advancement in TV programs to satellite launching [2]. One of the major areas that pertain to life globally is leadership, and this is why the issue of election is a major key in making decision to bring in the people's choice [3]. Most voting is done by traditional based voting. Many have lost interest in the traditional way of voting, because of injustice [4, 10].

The manual way of voting which has to do with balloting, has a lot of issues associated with it, e.g., the ballots may be hijacked [1]. Voting can be done twice by an individual if the ballot system of voting cannot detect whether the person is voting for the second time. A research was conducted some time ago on why voters do not come out to vote, and one major challenge common to them for not going to vote is because they will have to stay in the scorching sun or rain a times [5, 6] to be accredited as a valid voter and after which they will have to still queue after accreditation to vote. Figure 1 shows voters in queue in Australia and Figure 2 shows

cross section of voters in queue in Uganda, Africa, and most times it is not convenient.

Figure 1: Voters in Queue in Australia [5]

Figure 2: Voters in Queue in Uganda [6]

With the use of Electronic Voting System (EVS) it becomes easier to vote once you are eligible to vote and the system could take care of other activities done during and after the voting process [7, 8].

Another area of challenge with the manual process of election is the result counting stage. After voters are done with casting their votes, the election officials will open us the ballot box and begin to count the result, one after the other. This process takes much time for the results to be counted. The time varies based on population size and its accuracy may not be guaranteed, and manipulation can easily take place [9] with this kind of manual counting method, especially when it has to do with large number of voters in a polling unit, region, province, state or country. This also delays the result that should have been announced [10]. But, with the advent of EVS, all of these issues will be solved. There is a lot of issues associated with the manual way of voting even though many people still use it because they are not willing to make voting system void of rigging, double voting, ballot snatching, double count of results, than working with a system accuracy, efficiency, and security [11].

EVS have been practiced for the last three decades. In 1849, De Brettes initialized the first idea of electrical voting system (ElVS) based on decision making kind of telegraphs. In 1869, the first inventor of electrographic recorder was Thomas Edison [12]. In the ElVS which was introduced in1886 [13], the central recorder receives signals, and lists all the names of candidates in a matrix form using two columns with headings “YES” or “NO”.

1.2 The Initiative of Mechanical Voting

At the state of Victoria, Australia, election conducted manually using ballot first began in 1856, listing the names of candidates [14]. The first state who adopted this method in USA was Massachusetts (1888). As of 1889, January, punched card method was introduced to collect all data for the United States census by Herman Hollerith [13]. Later that same year, November 1889, New York patent on a

machine called Myers Automation booth, mechanical lever voting machine was issued. This machine helps preventing over-voting and also helps to speed- up the process of counting the votes. It was first used in 1892 in Lockport, New York [14]. Lever voting machine was also implemented in 1892 and was first used in New York [14].

Optical scan and Direct Recording Electronics (DRE) were introduced in early 1930-1985. In 1974, DRE was introduced; they are hardened physical machines which prevent access to connectors of PC like Universal Serial Bus. In Georgia, United States, as at early 1962, Punch Card Voting system was introduced using optical scanning voting system to read marked ballot and tally their corresponding results.

In 1990 – 2000, Internet voting (IV) was introduced and EVS became matured. Many developed countries are actually using EVS and IV method of voting ever since then till date, because it brought major solutions to many issues associated with manual voting system [15]. Nations like Switzerland, United Kingdom, Germany, India, Estonia and some developed nations are using this at the moment [16] instead of manual ways of voting. Systems have been built in various polling units and computer and mobile devices have also been used for voting [16]. At the moment, most of the states in the U.S. are using Remote electronic voting system called REVS through SERVE (Secure Electronic Registration and Voting Experiment) [17].

1.3 Electronic Voting Systems, Their Merits and Limitations

EVS is one of the most researched areas of interest today, because it has to do with information system security that could help make a difference from the ballot kind of voting having such problems as:

a. Much rigging during and after the election process
b. Voters could vote as many times as possible
c. During counting of votes manipulated figures are generated.
d. Queues cannot be contained and most people are discouraged to vote at polling booths.

Various studies and researches have been conducted in this area in the last 30 years [18]; many cryptographic methods have been employed with the aim of making it a secure EVS.

In [19], the following is stated. An issue associated with electronic voting is that before the signing of server, one can vote and when the server is signing, it amounts to double voting. This could be handled by blind signature, also cryptography can help to correct these errors associated with EVS. But, malicious users could want to produce a non-candidate signature for a crackdown of the system. Therefore, in [19], an EVS is proposed using a Novel Oblivious and Proxy Signature therein.

Chapter 2

LITERATURE REVIEW AND PROBLEM DEFINITION

In this chapter, firstly, we shall discuss on the history of paper ballot voting, the vulnerabilities and failure that is associated with EVS, cryptographic techniques used in EVS, and also the problem definition.

2.1 Brief History of Paper-Ballot Voting

In this section, we will discuss about the generations that voting was used. Voting system was first adopted by USA, and the first election held in 1789 [41], but election has undertaken various degree of transformation from one form to another. This transformation has been to paperless environment from paper-ballot system, from the manual system to technological system, from an offline base to an online base, and many much more [20]. We will start our discussion with the paper based ballot elections [21], people visits the nearby residential area where the polling unit is assigned to register, they will be given a voters identification card which contains their number and their information and after much confirmation is done, they are equally allowed to vote on the day of election. What they will just need to vote with is their cards [22]. This is a primitive way of voting, even though some countries and organization still practice this. Election managers or officials are trained to take care of the election processes on the registration stipulated time and the Election Day. The managers and officials are expected to be there with the materials for the voting (Ballot box, voters register, the ink pad for the thumb printing, the stamp, and election result materials) will be on ground at least an hour before the time of

election. This method can easily be understood. Also this method of voting has a lot of issues associated with it, like rigging, double voting, manipulation of result, cancellation of results, and very costly to manage [21].

2.2 The Vulnerabilities and Failures Associated with EVS

EVS uses computer to make voting process, simple and secretive, provide security, enable the counting process faster, reduce queuing at the voting centers, and can help those who are disability and the aged to exercise their franchise in voting, [22] have suggested across the nations of the world that voting with technology has much more issues than with the manual voting methods and results can easily manipulated. Others said; it is more expensive to implement EVS.

There has been a lot of issues accrue to EVS. In 2005, December, a company Black Box Voting Inc. made an investigation in Florida, within Leon County [23]. Two computer security experts Herbert Thompson and Harri Hursti made a huge headway by hacking EVS which used the central vote tabulator and they were able to manipulate the outcome of the election without anyone knowing that such an event known as hacking had taken place. This shows that there are software's packages that can be used to perform such hack events in EVS.

In [24], a report is given calling for a national database which shall be made publicly available, containing information on voting system failures and vulnerabilities. This report has found out that there are many maladies with EVS, many election managers and workers are not aware of this lacuna associated with electronic voting systems, because the programmers or the companies are not willing to tell the electorate the problem associated with their systems

In the last election that happened in America, between the two presidential candidate in 2016, Donald Trump (the incumbent as at today) and Hilary Clinton, it was said that is was the 58th presidential election that is held in America. The intelligence agencies of the United States government on 6th January 2017, asserted that the Russian government had interfered in the 2016 United States elections result [25, 26, 27]. And knowing fully well that the election was conducted electronically and it involves internet voting. President Trump has always showed his grievances against such claim by the intelligent unit stating that it is fake news. The president has also stated that the accusation against him (Trump campaign) and Russian collusion lacks proof and evidence.

With all of this in view, we will be introducing cryptography which will help minimize some aspect associated with EVS.

2.3 Cryptographic Techniques Used by EVS

Cryptography is the study and practice of techniques that is used for securing network communications in the midst of adversaries [28].

Davtyan et al complained that cryptography cannot give all the necessary protections to EVS [29].

We have two main classes of cryptographic techniques namely, Symmetric and Asymmetric cryptography [30].

2.3.1 Symmetric Cryptography

Symmetric-key cryptography [31] is defined as an encryption method that both the sender and receiver share the same key [30].

Symmetric (Secret key) algorithm like Data Encryption Standard (DES), demands that it is only possible to derive the secret key from the message that was encrypted alone. This is quantities of mathematical computations will involve doing so to make attempt that is infeasible with all a lot of hardware for the current computation [31]. Below, in Figure 3 is a diagram of symmetric cryptography.

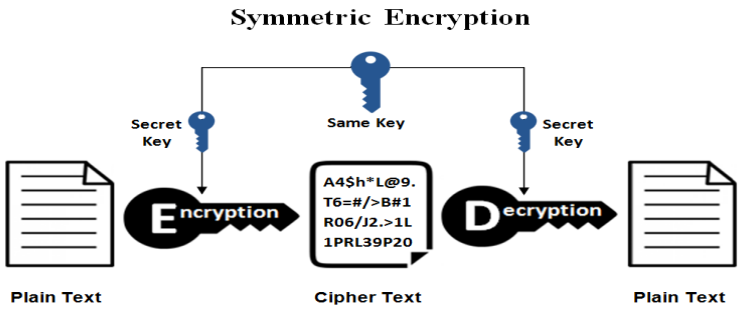


Figure 3: An Example of Symmetric Key Encryption [32]

Caesar cipher [30] is one of the oldest known simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals. First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ... , 'z'=25. We can now represent the Caesar cipher encryption function, e(x), where x is the character we are encrypting, as:

$$e(x) = (x + k) mod\ 26$$

Where k is the key (the shift) applied to each letter. After applying this function the result is a number which must then be translated back into a letter. The decryption function is:

$$d(x) = (x - k) mod\ 26.$$

2.3.2 Asymmetric Key Cryptography

Asymmetric key system or Public key system uses a pair of keys, each of which will decrypt the messages that were encrypted by another one, making sure one of these keys is kept secret (private key) [31].

RSA (Rivest–Shamir–Adleman) cipher is an example of Asymmetric key cryptography technique for decryption and encryption [30]. RSA is hard to be cracked that is based on the product of two large prime numbers factorization problem complexity. Below in Figure 4, is an overview of RSA.

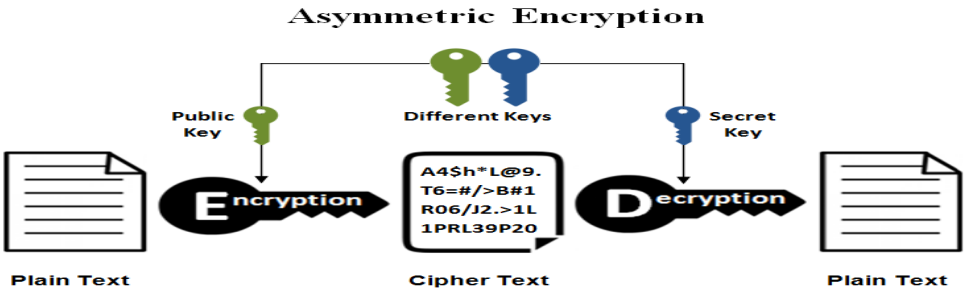


Figure 4: RSA - An Example of Asymmetric Key Cryptography [33]

RSA involves both public key and private key. The public key may be known by all; it is used to encrypt messages. A message that was encrypted using the public key will be decrypted only with private key. We are now ready to state the RSA scheme. The ingredients are the following:

p, q, two large prime numbers(private, chosen)

$n = p \cdot q$ (public, calculated)

e, with $\gcd(\varphi(n), e) = 1; 1 < e < \varphi(n)$ (public, chosen)

where

$$\varphi(n) = (p-1)(q-1))$$

$$d \equiv e^{-1} \bmod \varphi(n) \qquad \text{(private, calculated)}$$

The private key consists of {d,n}, and the public key consists of {e,n}[31]. Suppose that user A has published its public key and that user B wishes to send message M to A. Then B calculates $C = M^e \bmod n$ and transmits C. On receipt of this cipher text, user A decrypts by calculating $M = C^d \bmod n$. It is worthwhile to summarize the justification for this algorithm [27].

2.4 Novel Voting Protocol

Chen [35] in 1994, proposed oblivious signature. The schemes are of two types. The first is *n* messages with one key, a signee selects a message to be signed in order for the information not to be revealed to the signer, the second comprises of *n* keys with one message, a message is signed by the receiver with one out of *n* keys that is chosen by him or her, so that the signer is not aware of the key which was used by the receiver. Oblivious signature is contrast to blind signature, in that it makes sure the message signed is one of the predetermined messages; so that, the receiver cannot submit additional messages, the scheme will not accept the signature. Chou, also in 2012 [34], suggested another more secured and efficient *k* out of *n* oblivious scheme, for further clarity, Chiou in [36], gave more elaborate understanding on Novel *t* -out-of- *n* Oblivious Signature. The proposed protocol [19] uses the oblivious signature scheme that has the proxy protected and the proxy un-protected type. The scheme has four phases: 1), The System setup phase, 2), the proxy phase, 3), the signing phase, and 4) the verification phase in this protocol [19].

The scheme proposed for the signature consists of four entities: The original signer tagged as A, a proxy signer is tagged as B, a receiver is tagged as R, and a verifier is tagged as V. In this proposed scheme [19], let's say that the communications channel

among A and B is well secured. Any identity R or V communicates with B through insecure channel, making room for adversaries to intercept.

The proxy-unprotected protocol type and that of the proxy protected protocol are as follows:

2.4.1 Proxy-Unprotected Type Voting Scheme

A. System Setup Phase

This phase is also known as the key generation phase for both the original signer and the proxy signer, it helps generate two large prime numbers arranged with pair of keys, both for private and public for proceeding in the process. The scheme is represented by Steps 1-4 below,

Step 1. Two large primes p, q are chosen such that

$$q \mid (p - 1) \tag{2.1}$$

Step 2. Two values, g and h, from Z_p^* of order q, are chosen:,

$$g^q mod p = 1, h^q mod p = 1. \tag{2.2}$$

Step 3. The original signer, A, chooses a random number, $x_A \in Z_q^*$, as his private key, and computes his public key

$$y_A = g^{x_A} \bmod p . \tag{2.3}$$

Step 4. The proxy signer, B, chooses a random number, $x_B \in Z_q^*$, as his private key and computes his public key

$$y_B = g^{x_B} \bmod p . \tag{2.4}$$

We shall now use some numerical examples to test the authenticity of the system set up phase:

Example 1. According to (2.1). Let p $=$ 11, q $=$ 5 be our prime numbers chosen, $p-1=\ 11-1\ =10, 10/5=\ 2$ From equation (2.2), we have

$$h=5\ , g=9, Ord\ h=q,\quad Ord\ g=\ q\ \ =5$$

Table 1: Element h from $\boldsymbol{Z_p^*}$ of Order q

h^x\exponent, x	1	2	3	4	5
5^x	5	3	4	9	1

Table 2: Element g from $\boldsymbol{Z_p^*}$ of Order q

g^x\exponent x	1	2	3	4	5
9^x	9	4	3	5	1

From (2.3), we have p $=11$, $Z_q^*=\ \{1,2,3,4\}$.

Let $x_A=\ 2$, and $x_B=4$ then $y_A=g^{x_A}$ mod 11, $y_A=9^2 mod\ 11=4$.

From (2.4), we have $y_B=g^{x_B}$ mod $11=9^4 mod\ 11=5$.

End of Example 1.

B. Proxy Phase

This phase delegates authority from A, the original signer, to the proxy signer, B. It is represented by Steps 1-3 below:

Step 1. The original signer, A, selects a random value, $k\in_R Z_q^*$, and makes computations (2.5)-(2.7), illustrated by Figure 5:

$$r\ =\ g^{\,k} \bmod p \tag{2.5}$$

$$s_p=\ x_A r+\ k \bmod q \tag{2.6}$$

$$y_p\ =g^{s_p} \bmod p \tag{2.7}$$

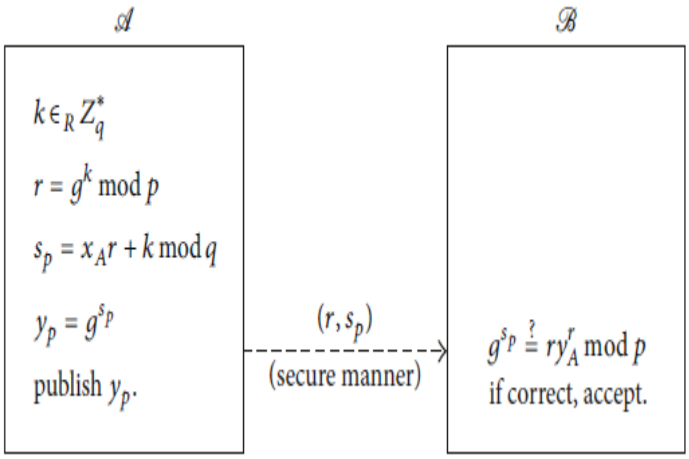


Figure 5: Proxy Phase [19]

Step 2. The original signer, A, securely sends the pair (r, s_p) to B and y_p is published.

Step 3. The proxy signer, B, verifies whether the values of left- and right-hand sides of

$$g^{s_p} = r y_A^r \, mod \, p \tag{2.8}$$

are equal to each other. If the values are equal, B accepts the proxy, and uses the value s_p received and computed in (2.6) as its secret proxy signature key.

We shall now use numerical examples to check the proxy phase

For (2.5) we choose $k = 3$ from Z_q^*, $r = 9^3 \, mod \, 11 = 3$

For (2.6) all values from (2.1), (2.2), (2.3) and (2.5),

$s_p = 2 \cdot 3 + 3 \bmod 5 = 4, (r, s_p) = (3,4)$

$$y_p = g^{s_p} \bmod p = 9^4 mod \, 11 = 5.$$

Original signer, A, in a proxy transmission securely forwards the pair (r, s_p), it is (3,4), in our case, to B, and then publishes y_p (in our case, $y_p = 5$).

For (2.8), B checks if it holds (For our numbers, $5 = g^{s_p} = ry_A^r = 3 \cdot 4^3 = 5 mod\ 11 = 5$), If it does, B accepts the proxy, then uses s_p as her or proxy secret signature key.

C. Signing Phase

Signing phase is illustrated by Figure 6 and is represented by Steps 1-4 below:

Step 1. The voter, R, considers a list of candidates, represented as messages $\{m_1, m_{2,} \ldots., m_n\}$ received from B, chooses $m_{b,}$ the candidate of his choice, and randomly chooses v from Z_q^*. Then, he computes:

$$c = g^v h^{-b} \text{mod p,} \tag{2.9}$$

and forwards c to B.

Step 2. The proxy signer, B, chooses n random numbers k_i from $Z_q^*, i = 1,2,..n$, and computes:

$$K_i = g^{k_i} \text{ mod } p \tag{2.10}$$

$$\delta_i = c(gh)^i mod\ p \tag{2.11}$$

$$\widehat{e_i} = H(m_i\ , K_i \delta_i\ mod\ p\ ) \tag{2.12}$$

$$\widehat{s_i} = k_i - s_p \hat{e}_i \text{mod q} \tag{2.13}$$

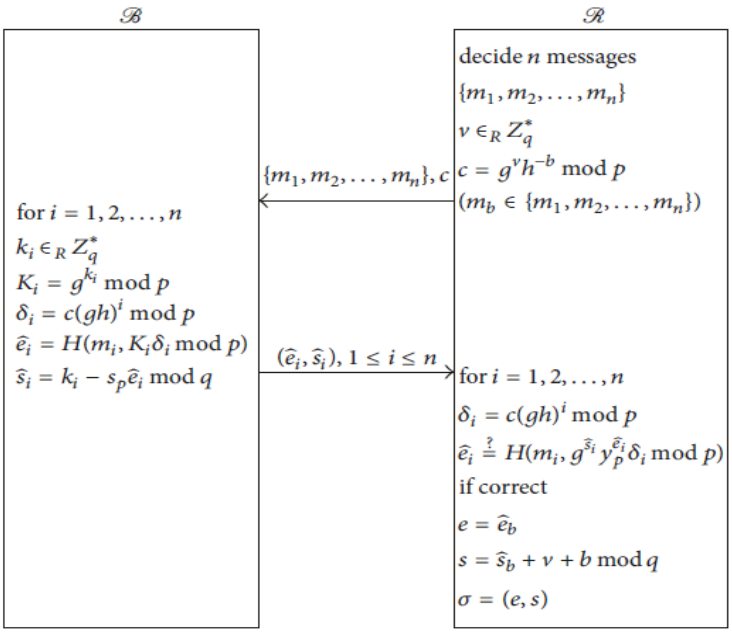


Figure 6: Signing Phase [19]

Then, B sends pairs $(\hat{e}_i, \hat{s}_i)$, $1 \le i \le n$, back to the voter, R

Step 3. For all i from 1 to n, the voter, R, computes (2.11) and accepts the oblivious signature if (2.14) holds:

$$\hat{e}_i == H\left(m_i, g^{\hat{s}_i} y_p{}^{\hat{e}_i} \delta_i \bmod p\right), \tag{2.14}$$

otherwise, the message is rejected.

Step 4. The voter, R, sets

$$e = \hat{e}_b \tag{2.15}$$

$$s = \hat{s}_b + v + b \bmod q. \tag{2.16}$$

The signature for the message, m_b, selected by R is

$$\sigma = (e, s). \tag{2.17}$$

The signature (2.17) together with m_b, is sent to the voting center, V, as shown in Figure 7.

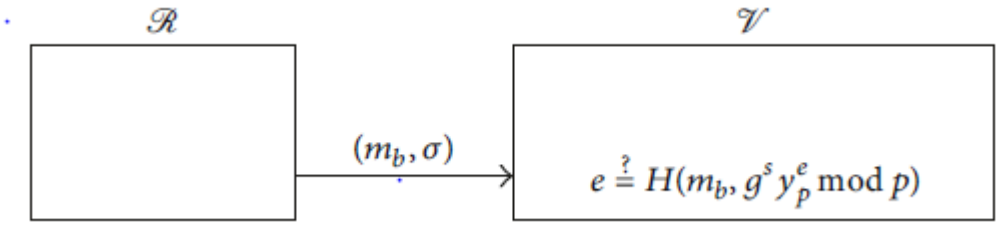


Figure 7: Verification Phase [19]

For our numerical examples for signing phase:

n=6,

$\{m_1, m_{2,} \ldots., m_n\} = \{10, 3, 2, 1, 4, 7\}, \mathrm{b} = 5, \mathrm{v} = 2,$

$c = g^v h^{-b} mod\ p = 9^2\ 5^{-5} \bmod 11 = 9^2 \cdot 3125^{-1}\ mod\ 11 = 9^2 \cdot 1\ mod\ 11 = 4$

k_i are choosen randomly :{1,2,3,4,1,2}

Let in (2.10), $k_1 = 1,\ k_2 = 2,\ k_3 = 3,\ k_4 = 4,\ k_5 = 1,\ k_6 = 2$. Then,

$K_1 = 9^1\ mod\ 11 = 9, K_2 = 9^2\ mod\ 11 = 4, K_3 = 9^3\ mod\ 11 = 3,$

$K_4 = 9^4\ mod\ 11 = 5, K_5 = 9^1\ mod\ 11 = 9, K_6 = 9^2\ mod\ 11 = 4,$

From (2.11),

$\delta_1 = 4(9 \cdot 5)^1 mod\ 11 = 4, \delta_2 = 4(9 \cdot 5)^2 mod\ 11 = 4,$

$\delta_3 = 4(9 \cdot 5)^3 mod\ 11 = 4, \delta_4 = 4(9 \cdot 5)^4 mod\ 11 = 4,$

$\delta_5 = 4(9 \cdot 5)^5 mod\ 11 = 4, \delta_6 = 4(9 \cdot 5)^6 mod\ 11 = 4,$

From (2.12),

$$\widehat{e_i} = H(m_i\ , K_i \delta_i\ mod\ p\ ).$$

The hash function is be defined as follows. It takes the value of m_i , as a character string, concatenates it with a character string representing $K_i \delta_i\ mod\ p$, returns sum of ASCII codes of the characters modulo 256.

$$\widehat{e_1} = H(10\ ,9 \cdot 4\ mod\ 11\ ) = H(10,3) = 103 = 49 + 48 + 51 = 148,$$

$$\widehat{e_2} = H(3\ ,4 \cdot 4\ mod\ 11\ ) = H(3,5) = 35 = 51 + 53 = 104,$$

$$\widehat{e_3} = H(2,3 \cdot 4\, mod\ 11) = H(2,1) = 21 = 50 + 49 = 99,$$

$$\widehat{e_4} = H(1,5 \cdot 4\, mod\ 11) = H(1,9) = 19 = 49 + 57 = 106,$$

$$\widehat{e_5} = H(4,9 \cdot 4\, mod\ 11) = H(4,3) = 43 = 52 + 51 = 103,$$

$$\widehat{e_6} = H(7,4 \cdot 4\, mod\ 11) = H(7,16) = 716 = 55 + 49 + 54 = 158.$$

From (2.13), recalling (2.6), (2.10),

$$\widehat{s_1} = 1 - 4 \cdot 148 \bmod 5 = 4$$

$$\widehat{s_2} = 2 - 4 \cdot 104 \bmod 5 = 1$$

$$\widehat{s_3} = 3 - 4 \cdot 99 \bmod 5 = 2$$

$$\widehat{s_4} = 4 - 4 \cdot 106 \bmod 5 = 0$$

$$\widehat{s_5} = 1 - 4 \cdot 103 \bmod 5 = 4$$

$$\widehat{s_6} = 2 - 4 \cdot 158 \bmod 5 = 0$$

For i = 1, 2, . . , n, R computes and accepts the oblivious signature if and only if (2.14) holds,

$$148 = \widehat{e_1} = H(10,\ 9^4 \cdot 5^{148} \cdot 4\, mod\ 11) = H(10,3) = 103 = 49 + 48 + 51 = 148,$$

$$104 = \widehat{e_2} = H(3,\ 9^1 \cdot 5^{104} \cdot 4\, mod\ 11) = H(3,5) = 35 = 51 + 53 = 104,$$

$$99 = \widehat{e_3} = H(2,\ 9^2 \cdot 5^{99} \cdot 4\, mod\ 11) = H(2,1) = 21 = 50 + 49 = 99,$$

$$106 = \widehat{e_4} = H(1,\ 9^0 \cdot 5^{106} \cdot 4\, mod\ 11) = H(1,9) = 19 = 49 + 57 = 106,$$

$$103 = \widehat{e_5} = H(4,\ 9^4 \cdot 5^{103} \cdot 4\, mod\ 11) = H(4,3) = 43 = 52 + 51 = 103,$$

$$158 = \widehat{e_6} = H(7,\ 9^0 \cdot 5^{158} \cdot 4\, mod\ 11) = H(7,16) = 716 = 55 + 49 + 54 = 158$$

For (2.15)-(2.17), from (2.12), (2.13), $v = 2, b = 5,\ e_5 = 103, s_5 = 4,$

$e = 103,\ s = 4 + 2 + 5\, mod\ 5 = 1.$

Values for (2.17):

$\sigma = (103,1)$. So the signature is valid based on the computations.

D. Verification Phase

Verification phase is illustrated by Figure 7 and represented by Step 1 below:

Step 1. The voting center, V, accepts the signature. σ, when (2.18) holds:

$$\text{e} == \text{H}\ (m_b, g^s y_p^e \bmod \text{p}). \tag{2.18}$$

We shall now use numerical examples to test our verification stage:

For our numerical examples for the verification stage of the scheme [19]

From (2.14), (2.15), (2.16), we verify (2.18) using values:

$$103 = e = H\ (4, 9^1 \cdot 5^{103}\ mod\ 11) = H(4,3) = 43 = 52 + 51 = 103.$$

This shows that the signature is valid for the fifth message chosen by R, the voter.

2.4.2 Proxy-Protected Type Scheme

Signing and verification phase of the proxy-protected type are the same as that of the proxy unprotected, the only distinguishing factor between them is that for the proxy protected one it has an additional mathematical computation

$$s_p = s_A + x_B \bmod \text{q} \tag{2.19}$$

as illustrated by Figure 8.

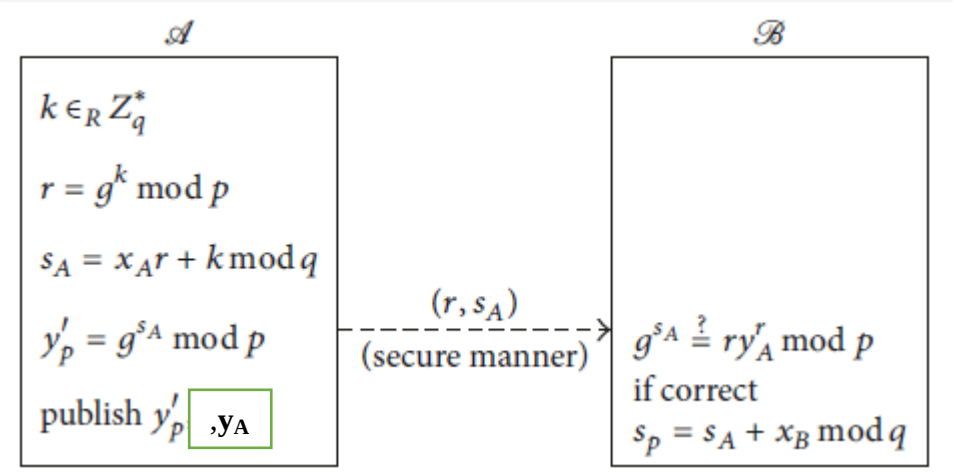


Figure 8: Proxy Protected [19]

Publishing of $\boldsymbol{y_A}$ is added by us in Figure 8 (according to Section 3.4).

Proxy-Protected- Phase

The proxy-protected phase is represented by Steps 1-3 below:

Step 1. The original signer, A chooses k that belongs to Z_q^* randomly, makes $r = g^k \bmod p$ as in (2.5) and then computes

$$s_A = x_A r + k \ mod \ q \tag{2.20}$$

$$y_p' = g^{s_A} \ mod \ p \tag{2.21}$$

The proxy transmission takes place after the computation is done.

Step 2. The original signer A securely sends the pair,(r, s_A) to B and y_p' is published.

Step 3. The proxy signer B checks whether

$$g^{s_A} = r y_A^r mod \ p \tag{2.22}$$

holds. If it is does, B accepts the proxy and computes its secret proxy signature key.

$$s_p = s_A + x_B \bmod q \tag{2.23}$$

2.4.3 Voting Protocol Design and Implementation

In the proposed in [19] voting system, the same actors are expected as for the schemas from Sections 2.4.1, 2.4.2: the creator (central government) A delegates authority to proxy signer (local government) B, and R voter, can get a legal ballot from B and send his or her vote to V (voting center), and vote results are displayed on the bulletin BB. This protocol design has six phases. They are mostly similar to those considered in Sections 2.4.1, 2.4.2 but introduce some new details related with RSA, registration, and votes counting. The phases, system setup, proxy, registration, circling, voting, and counting, is shown in Figure 9, and the voting protocol computation time measured in (milliseconds), is shown in Table 4.

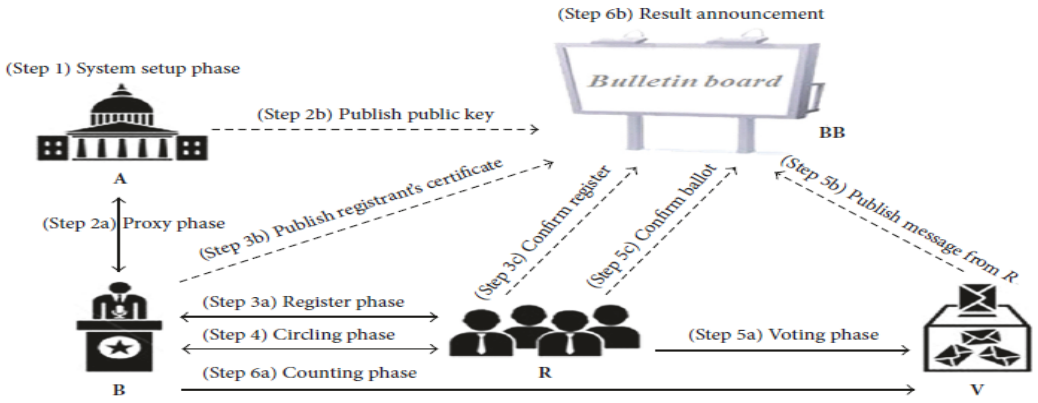


Figure 9: System actors and phases [19]

The system involves six phases represented by Steps 1-6 below

Step 1. **Setup phase**: parameters are generated.

Step 2. Proxy phase

Step 2a. The original signer, A, delegates authority to B, the proxy signer.

Step 2b. Proxy signer, B, then publishes the public key to the bulletin.

Step 3. Register phase

Step 3a. Proxy signer, B, checks whether R, voter, is legally registered; if so, a voting certificate is issued to R.

Step 3b. Proxy signer, B, publishes all the certificates to the bulletin.

Step 3c. Voter, R, checks via the bulletin whether he is registered successfully.

Step 4. Circling phase: The voter chooses a candidate and receives the signature on it from the proxy signer.

Step 5. Voting phase

Step 5a. Voter, R, casts his vote by sends it to the voting center.

Step 5b. Voting center, V, immediately publishes a message about the vote cast by R to BB.

Step 5c. Every voter, R, can confirm whether his or her ballot has been received by the voting center; if not, he or she can resend the ballot.

Step 6. Counting phase

Step 6a. When the voting period has ended, B forwards the decrypting key to V, and V verifies and counts the votes.

Step 6b. V publishes the result of the votes to BB, where every voter can count and verify all votes.

2.4.4 Processes Involved in the Voting System, Experimental Settings and Results

We assume that the system database already contains list of voters, and that BB is read-only to all entities.

2.4.4.1 System Setup Design

This phase is represented by Steps 1-9 below [19].

Step 1. Two large primes p, q are selected according to (2.1).

Step 2. Two generators, g, h, belong to Z_p^* of order q are chosen, according to (2.2).

Step 3. The original signer A chooses a random number, $x_A \in Z_q^*$ as his private key to compute the public key according to (2.3), A publishes p, q, g, h, and y_A on BB.

Step 4. The proxy signer B also chooses a random number, $x_B \in Z_q^*$ as his private key to compute the public key

Step 5. B chooses two large primes p_B, q_B

Step 6. B computes

$$N_B = p_B \cdot q_B, \tag{2.24}$$

Step 7. B computes totient function, $\emptyset(N_B)$ as

$$\emptyset(N_B) = (p_B - 1)(q_B - 1), \quad (2.25)$$

Step 8. B chooses RSA public key, e_B, such that

$$\mathrm{GCD}(e_B, \emptyset(N_B)) = 1. \quad (2.26)$$

Step 9. B computes RSA private key

$$d_B = e_B^{-1} \bmod \emptyset(N_B), \quad (2.27)$$

Proxy signer, B, publishes N_B, e_B, and y_B on BB.

Numerical data can be used for the voting protocol exactly as in Example 1.

Example 2. Let $p_B = 11$, $q_B = 5$.

According to (2.24), $N_B = 11 \cdot 5 = 55$.

According to (2.25), $\emptyset(N_B) = (11 - 1)(5 - 1) = 40$.

According to (2.26), $\mathrm{GCD}(e_B, 40) = 1$, $e_B = 3$.

According to (2.27), $d_B = 3^{-1} \bmod 40 = 27$.

Now, we need to compute $d_B = e_B^{-1} \bmod \emptyset(N_B)$ by using backward substitution of GCD algorithm:

According to GCD:

$$40 = 3 \cdot 13 + 1$$

$$13 = 1 \cdot 13 + 0$$

Therefore, we have:

$$1 = 40 - 3 \cdot 13$$

Hence, we get $d_B = e_B^{-1} \bmod \emptyset(N_B) = e_B^{-1} \bmod 40 = -13 \; mod \; 40 =$

$(27 - 40) \; mod \; 40 = 27$

So, the public key is {3, 55} and the private key is {27, 55}

End of Example 2.

2.4.4.2 Proxy Phase Design

Proxy phase is illustrated by Figure 10 and represented by Steps 1-5 below.

Step 1. Original signer, A, randomly chooses k that belongs to Z_q^* and calculates

$$r_A = g^k \bmod p\,, \tag{2.28}$$

$$s_A = x_A r_A + k \bmod q\,, \tag{2.29}$$

$$y_p' = g^{s_A} \bmod p, \tag{2.30}$$

the same as (2.5, 2.20, 2.21).

Step 2. Original signer, A, RSA encrypt the pair of (r_A, s_A) using (e_B, N_B) , then sends it to B.

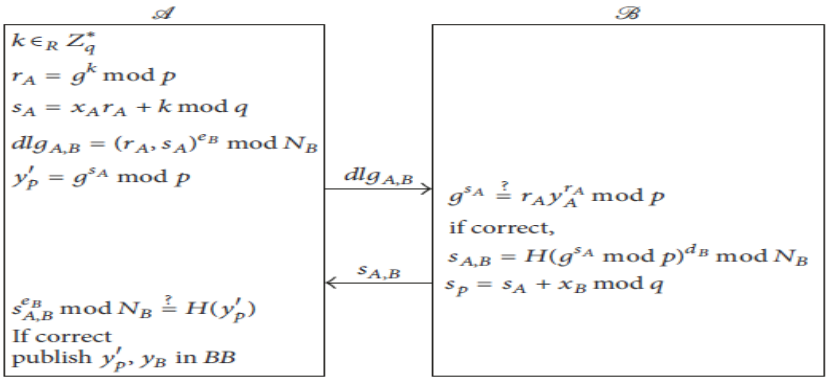


Figure 10: Proxy phase design [19]

Step 3. B with RSA, decrypts (r_A, s_A) using (d_B, N_B) , and checks whether equality

$$g^{s_A} == r_A y_A^{r_A} mod\ p \tag{2.31}$$

is true, similar to (2.8). If it is true, B accepts the proxy and calculates

$$s_p = s_A + x_B\ mod\ q \tag{2.32}$$

as the secret proxy signature key.

Step 4. The proxy signer, B, generates a signature

$$S_{A,B} = H(g^{s_A}\ mod\ p)^{d_B} \bmod N_B \quad , \tag{2.33}$$

and forwards it to A.

Step 5. A checks whether

$$s_{A,B}^{e_B} == H\,(y_p')\bmod N_B \tag{2.34}$$

is true. If it is true then y_p', y_B published by A to the bulletin board BB.

Example 3. From (2.28),(2.29),(2.30)

$$r_A = 9^3 mod\ 11 = 3.$$

$$\mathrm{s_A} = 2 \cdot 3 + 3 \bmod 5 = 4,$$

$$(r_A, s_A) = (3,4),$$

$$y_p' = g^{s_A} \bmod \mathrm{p} = 9^4 mod\ 11 = 5,$$

The original signer, A, encrypts the pair $(r_A, s_A) = (3,4)$ using $(e_B, N_B) = (3, 55)$, result of the encryption :$(r_A = 3, s_A = 4), (e_B = 3, N_B = 55)$,

encryption of $r_A = 3^3 mod\ 55 = 27$, encryption of $s_A = 4^3 mod\ 55 = 9$, the pair is (27,9) and publishes $y_p' = 5$

The proxy signer, B, RSA decrypts the message received using $(d_B, N_B) = (27,55)$, decryption of message :$(27, 9), (d_B = 27, N_B = 55)$,

decrypt $r_A = 27^{27} mod\ 55 = 3$, decrypt $s_A = 9^{27} mod\ 55 = 4$, the pair is (3,4),B checks equality of (2.31) , for our numbers, $5 = g^{s_A} = r y_A^r = 3 \cdot 4^3 = 5\ mod\ 11 = 5$, which holds, and B accepts it and computes $s_p = 4 + 4\ mod\ 5 = 3$, according to (2.32),

From (2.33) and (2.34)

$$S_{A,B} = H(9^9\ mod\ 11)^{27} \bmod 55 = \mathrm{H}(5)^{27} \bmod 55 = 53^{27}\ mod\ 55 = 37,$$

$$s_{A,B}^{e_B} = 37^3 mod\ 55 = \mathrm{H}\,(5) = 53,$$

Note that ASCII code for '5' is 53.

$$H\,(y_p') \bmod N_B = H(5) mod\ 55 = H(5) = 53.$$

End of Example 3.

2.4.4.3 Register Phase Design

Register phase is illustrated by Figure 11 and is represented by Steps 1-3 below:

Step 1. Voter, R, picks pn as a pseudo-name and a password, pw, computes

$$H_R = \text{H (pw)}, \tag{2.35}$$

RSA encrypts (id, pn,(pw)) using (e_B,N_B), and R sends it to B, where id is the voter's identification number.

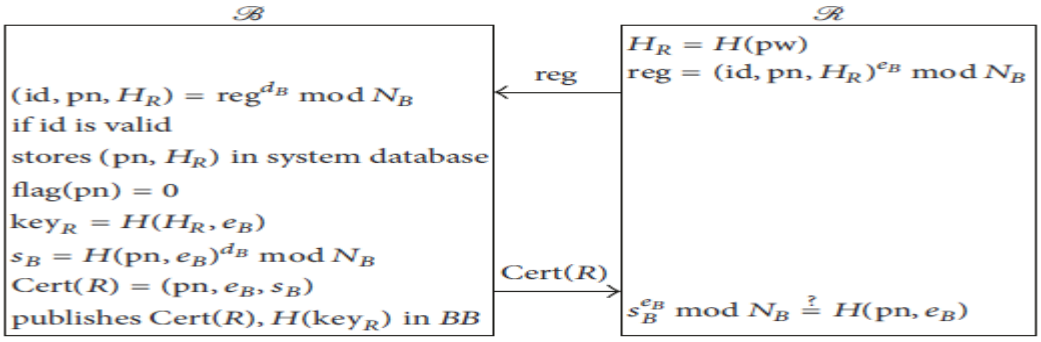


Figure 11: Register phase design [19]

Step 2. The proxy signer, B, decrypts (id, pn, H_R) using (d_B, N_B) to check whether R the voter is a legal voter. If R is, then, B stores (pn, H_R) in the system database, sets flag $(pn) = 0$, and calculates

$$key_R = H(H_R, e_B), \tag{2.36}$$

calculates RSA signature of (pn, e_B),

$$s_B = H(pn, e_B)^{d_B} \bmod N_B\,, \tag{2.37}$$

returns Cert(R) to R, and the Cert(R) is published to BB, where

$$\text{Cert(R)} = (\text{pn}, e_B, s_B). \tag{2.38}$$

Step 3. Then, R checks whether

$$s_B^{e_B} = \text{H}\,(\text{pn}, e_B) \bmod N_B \tag{2.39}$$

is true. Note that (2.39) is not proved in [19]. If it is true, R has the right to vote.

Example 4 illustrates below numerically the register phase.

Example 4. According to (2.35), R enters password, pw= "FEYI", and pseudo-name, pn="OLA".

ASCII codes for each letter are added modulo N:

$$\mathrm{H_R} = \mathrm{H(pw)} = 70 + 69 + 89 + 73 = 301 \bmod 55 = 26$$

Encryption is done for each of id, pseudo-name and password, i.e., $(id, pn, H(pw))$ $(e.g., = (2, OLA, 26))$ using $(e_B, N_B) = (3, 55))$. For encryptionof the pseudo-name, pn, we introduce character coding Table 3:

Table 3: Character coding table

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

We now encrypt the value of each character. For OLA: 16-13-2, this takes each value before the dash and uses an RSA encryption as follows: we used $(e_B, N_B) = (3, 55))$ as our encryption keys, we have $16^3 mod\ 55 = 26,\ 13^3 mod\ 55 = 52, 2^3 mod\ 55 = 8$ after these encryption, we have $(8, 26{,}52{,}8, 31)$.

The encrypted $(id, pn, H(pw))$ is decrypted by B to give us $(2, OLA, 26)$ then R is checked to be a legal voter and the flag (pn) value is set to be zero.

From (2.36), e.g., $key_R = \mathrm{H}\ (45{,}3) = 52 + 53 + 51 \bmod 55 = 46$

From (2.37), $\mathrm{s_B} = \mathrm{H(OLA, 3)}^{27} \bmod 55 = (\ \mathrm{H(OLA||3)} = 79 + 76 + 65 + 51 = 271 \bmod 55 = 51 = \mathrm{s_B} = (51)^{27} \bmod 55 = 6$

The certificate, $\mathrm{Cert(R)} = (\mathrm{pn}, e_B, \mathrm{s_B}) = (\mathrm{OLA}, 3{,}6)$, is forwarded to R.

Verification is done by voter, R, according to (2.39):

$s_B^{e_B} = 6^3 \bmod 55 = 51$

H (OLA, 3) mod $55 = 79 + 76 + 65 + 51 = 271 \bmod 55 = 51$, and it is verified because (2.39) holds.

End of Example 4.

2.4.4.4 Circling Phase

Circling phase is illustrated by Figure 12 and is represented by Steps 1-4 below:

Step 1. The proxy signer, B, forwards a random number r belonging to Z_q^* to voter R; this is done after receiving a login request from R, as illustrated in Figure 12.

Step 2. The voter, R, computes

$$H_R' = \text{H(H(pw), r)} \tag{2.40}$$

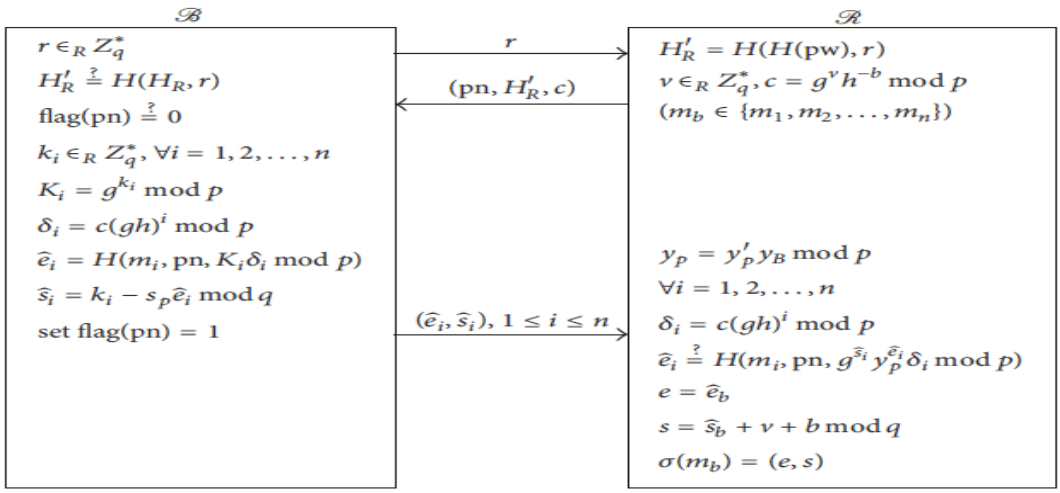


Figure 12: Circling phase design [19].

then R picks a random number *v* belonging to Z_q^* , and computes

$c = g^v h^{-b} \bmod \text{p}$ according to (2.9), $(m_b \in \{m_1, m_2, \ldots, m_n\})$, and forwards (pn, H_R', c) to B.

Step 3. The proxy signer, B, examines whether

$$H_R' = \text{H}(H_R, \text{r}) \tag{2.41}$$

is correct. If so, B checks whether $flag(pn) = 0$. If true, B chooses

$k_i \in_R Z_q^*$, calculates $K_i = g^{k_i} \bmod p$,according to (2.10), $\delta_i = c(gh)^i mod\ p$

,according to (2.11), $\widehat{e_i} = H(m_i, pn, K_i\delta_i\ mod\ p)$

according to (2.12), and

$$\widehat{s_i} = k_i - s_p\hat{e}_i \bmod q$$

according to (2.13), $\forall i$ =1, 2, . . . , n, returns $(\hat{e}_i, \widehat{s_i})$, $1 \le i \le n$, to R, and sets flag(pn) =1.

Step 4. R computes

$$y_p = y_p' y_B \bmod p ,$$

and, for every $i = 1, 2, \ldots, n$, calculates $\delta_i = c(gh)^i mod\ p$, according to (2.11), and checks whether

$$\widehat{e_i} == H\left(m_i, pn, g^{\widehat{s_i}}\ {y_p}^{\widehat{e_i}}\delta_i\ mod\ p\right) \qquad (2.42)$$

is correct. If so, R computes

$$s = \widehat{s_b} + v + b \bmod q ,$$

according to (2.16), and

$$e = \widehat{e_b},$$

according to (2.15).

The final signature is

$$\sigma\ (m_b) = (e, s),$$

according to (2.17).

Note that (2.42) is not proved in [19]. The signing phase is illustrated numerically by Example 5 below.

Example 5. The proxy signer, B, sends a random chosen number $r = 3 \in_R Z_q^* =$ {1,2,3,4} to R, after which the voter is requested to login.

The voter R does some computations from (2.40)

$H'_R = H(H(pw), r) = H(45,3) = 156 \bmod 55 = 46$ for hash calculation.

Also, from (2.9), for our values, $n = 6, \{m_1, m_{2,} \ldots, m_n\} = \{10,3,2,1,4,7\}, b = 5,$

$v = 2, c = g^v h^{-b} \bmod p = 9^2\, 5^{-5} \bmod 11 = 9^2 \cdot 3125^{-1} \bmod 11 =$

$(3125^{-1} \bmod 11 = 1,$ this can be rewritten as, $3125 \cdot x = 1 \bmod 1, then\ x = 1) = 9^2 \cdot 1 \bmod 11 = 4,$

$(m_b \in \{m_1, m_{2,} \ldots, m_n\})$, he then forwards the message $(pn, H'_R, c) =$ $(OLA, 46,4)$ to B.

From (2.41), $H'_R = H(H_R, r) = H(45,3) = 156 \bmod 55 = 46$

From (2.10),

$k_1 = 1,\ k_2 = 2,\ k_3 = 3,\ k_4 = 4,\ k_5 = 1,\ k_6 = 2,$

$K_1 = 9^1 \bmod 11 = 9, K_2 = 9^2 \bmod 11 = 4, K_3 = 9^3 \bmod 11 = 3, K_4 = 9^4 \bmod 11 = 5, K_5 = 9^1 \bmod 11 = 9, K_6 = 9^2 \bmod 11 = 4$

From (2.11),

$\delta_1 = 4(9 \cdot 5)^1 mod\ 11{=}4,\ \delta_2 = 4(9 \cdot 5)^2 mod\ 11{=}\ 4,\ \delta_3 = 4(9 \cdot 5)^3 mod\ 11{=}4$

$\delta_4 = 4(9 \cdot 5)^4 mod\ 11{=}4,\ \delta_5 = 4(9 \cdot 5)^5 mod\ 11{=}4,\ \delta_6 = 4(9 \cdot 5)^6 mod\ 11{=}4$

Using a hash function that has three inputs introduced in (2.14).

$$\widehat{e_i} = H(m_i, pn, K_i \delta_i \bmod p), \qquad (2.43)$$

$\widehat{e_1} = H(10, OLA, 9 \cdot 4 \bmod 11) = H(10, OLA, 3) = 49 + 48 + 79 + 76 + 65 + 51 \bmod 55 = 38$

$\widehat{e_2} = H(3, OLA, 4 \cdot 4 \bmod 11) = H(3, OLA, 5) = (51 + 79 + 76 + 65 + 53) mod\ 55 = 49$

$\widehat{e_3} = H(2, OLA, 3 \cdot 4 \bmod 11) = H(2, OLA, 1) = 50 + 79 + 76 + 65 + 49\ mod 55 = 44$

$\widehat{e_4} = H(1, OLA, 5 \cdot 4 \, mod \, 11) = H(1, OLA, 9) = 49 + 79 + 76 + 65 +$

$57 mod \, 55 = 51$

$\widehat{e_5} = H(4, OLA, 9 \cdot 4 \, mod \, 11) = H(4, OLA, 3) = 52 + 79 + 76 + 65 +$

$51 \, mod 55 = 48$

$\widehat{e_6} = H(7, OLA, 4 \cdot 4 \, mod \, 11) = H(7, OLA, 5) = 55 + 79 + 76 + 65 +$

$53 \, mod \, 55 = 53$

From (2.13), recalling (2.2), (2.10),

$$\widehat{s_1} = 1 - 3 \cdot 38 \bmod 5 = 2,$$

$$\widehat{s_2} = 2 - 3 \cdot 49 \bmod 5 = 0,$$

$$\widehat{s_3} = 3 - 3 \cdot 44 \, mod \, 5 = 1,$$

$$\widehat{s_4} = 4 - 3 \cdot 51 \, mod \, 5 = 1,$$

$$\widehat{s_5} = 1 - 3 \cdot 48 \, mod \, 5 = 2,$$

$$\widehat{s_6} = 2 - 3 \cdot 53 \, mod \, 5 = 3$$

. $$(\hat{e}_i, \widehat{s_i}) = (38,2), (49,0), (44,1), (51,1), (48,2), (53,3)$$

$$s = 2 + 2 + 5 \, mod \, 5 = 4$$

according to (2.17), $\sigma(m_5) =$ (e,s) =(48,4)

End of Example 5.

2.4.4.5 Voting Phase

Voting phase is illustrated by Figure 13 and is represented by Steps 1-3 below:

Step 1. The voter R calculates $H(H(pw), e_B)$ and uses a Caesar cipher symmetric key encryption to encrypt $(m_b, \sigma(m_b))$ to generate a cipher C_R, and sends $(Cert(R), C_R)$ to the voting center,V.

Step 2. The voting center, V, first examines whether

$$s_B^{e_B} = H(\text{pn}, e_B) \bmod N_B$$

according to (2.39), illustrated in Figure 13, holds. If so, V publishes $(Cert(R), C_R)$ to BB.

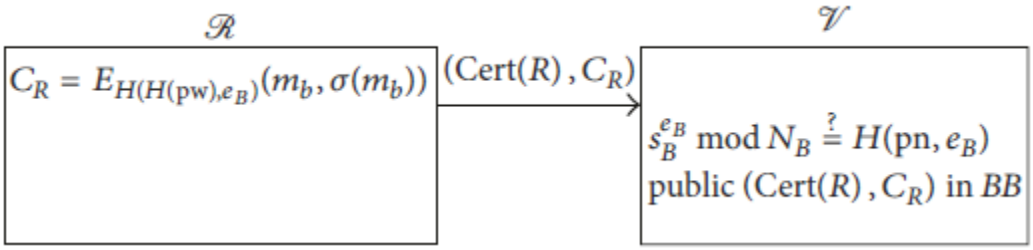


Figure 13: Voting phase design [19]

Step 3. Every voter R can check if their vote is received by V via BB. If not, R resends $(Cert(R), C_R)$.

The voting phase is illustrated numerically by Example 6 below continuing Example 5.

Example 6. R uses a symmetric key $key_R = H(45,3) = 46$ to encrypt $(m_b, \sigma(m_b))$ =([4], [48,4]), using Caesar cipher encryption; $c = p + k \bmod N$, where c is cipher, p is plaintext, and k is the key (the shift), therefore, we have $4 + 46\, mod\ 55 = 50$, $48 + 46\, mod 55 = 39$, 4+46 mod55=50; which produces $C_R = (50,39,50)$. It will now send $(Cert(R), C_R)$ as ([$OLA, 3, 5$]), ([50,39,50]) to the voting center, V.

The voting center, V checks if

$s_B^{e_B} = 6^3 \bmod 55 = 51 = H(OLA, 3) \bmod 55 = 79 + 76 + 65 + 51\, mod 55 = 51$

is true, V publishes (Cert(R), C_R) on BB and each voter can check its presence there, End of Example 6.

2.4.4.6 Counting Phase

Counting phase is illustrated by Figure 14 and is represented by Steps 1-3 below:

Step 1. The proxy signer B forwards key,

$$key_R = H(H_R, e_B)\ , \tag{2.44}$$

to V.

Step 2. The voting center, V, uses a Caesar cipher to decrypt C_R using the symmetric key, key_R, publishes $(Cert(R), C_R, m_b, key_R)$ to BB, and calculates

$$y_p = y_p' y_B \bmod p,$$

and verifies whether

$$e == H(m_b, pn, g^s y_p^e \bmod p) \tag{2.45}$$

is true. If (2.45) is valid, the vote is counted.

Step 3. The voting center, V, publishes the voting results on BB. Everyone can count and verify the ballots from BB.

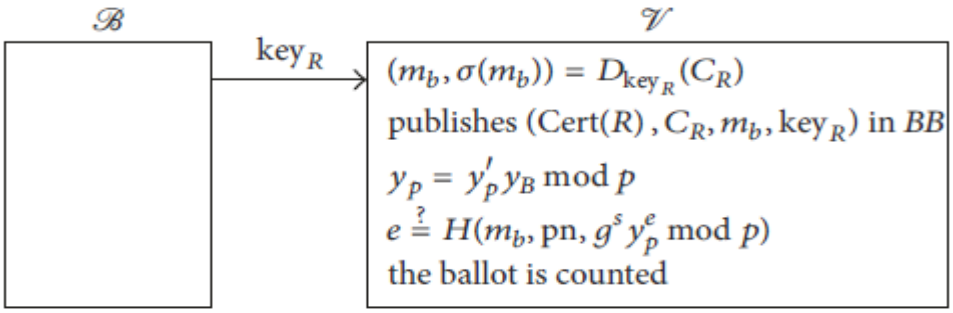


Figure 14: Counting phase design [19]

The counting phase is illustrated numerically by Example 7 below.

Example 7. The proxy signer, B, forwards the key, $key_R = \text{H}(45,3) = 46$, to V. The voting center decrypts using symmetric encryption, $\text{C}_\text{R} = ([50,39,50])$ using $key_R = 46$, with formula p = c - k mod N, where c is cipher, p is plaintext and k is key, therefore we have 50-46=4, 39-46=-7+55=48, 50-46=4; we have this result ([4], [48,4]), then it publishes $(Cert(R), C_R, m_b, key_R) = ((OLA, 3,5)\ , (4,48,4),4,46)$,

From (2.18), also adding a final variable

$$y_p = y'_p y_B \text{mod p} = 5 \cdot 5 \text{ mod } 11 = 3$$

$48 = \text{e} = \text{H}(m_b, pn, g^s y_p^e \text{mod } p) = \text{H}(4, \text{OLA}, 9^4 \cdot 3^{48} mod\ 11) = H(4, OLA, 3) =$

$(52 + 79 + 76 + 65 + 51) mod\ 55 = 48$, which is correct.

End of Example 7.

2.4.4.7 Experimental Settings and Results

Table 4 below shows the computation time for each of the phases involved in the proposed voting system [19] measured in milliseconds. For the voting protocol [19], it was not specified the software package that was used for its implementation but the results were given, where PP stands for the Proxy Phase, RP stands for the Registration Phase, CiP stands for the Circling Phase, VP stands for the Voting Phase and CoP stands for the Counting Phase. It gives the computational role for each phase.

Table 4: Computation time of the voting protocol (milliseconds) [19] phases PP, RP, CiP, VP, and CoP, and roles A, B, R, and V.

Phase	PP		RP		CiP		VP		CoP
Role	A	B	R	B	R	B	R	V	V
Time	42.4	31.5	22.85	19.55	31.2	31	19.8	10.35	20.25

2.5 Problem Definition

The problem definition and the aims of the thesis are:

1. Study the VS [19].
2. Provide proofs for oblivious signature and VS [19] formulas (2.8), (2.14), (2.18), (2.34), (2.39), and (2.42).

3. Design, implement and test VS [19] to ascertain the scheme proposed just for a single user
4. Conduct experiments with the VS similar to those made in [19] and compare them versus Table 4.

The rest of the thesis is organized as follows. In Chapter 3, we give proofs of the conditions used in VS [19] without justification, and adjust VS [19] according to inconsistences found. In Chapter 4, we discuss our VS [19] design, implementation and testing. In Chapter 5, we conduct experiments in the settings of [19] and compare our results versus Table 4. In Chapter 6, we conclude the thesis and discuss future work.

Chapter 3

ANALYSIS AND PROOF OF THE VOTING SYSTEM CORRECTNESS

We analyze the VS [19] described in Chapter 2 and prove consistency of the conditions (2.8), (2.14) and (2.18) used in the novel oblivious signature and (2.34), (2.39), and (2.42) used in VS [19] without justification. The VS is adjusted according to the proofs made.

3.1 Proof of the Conditions Used in Novel Oblivious Signature Scheme [19]

We prove conditions (2.8), (2.14) and (2.18) used in VS [19].

3.1.1 Proof of (2.8)

Let us prove that (2.8) holds.

Proof. Consider its left-hand side (LHS). From (2.6), we get:

$$g^{s_p} = g^{x_A r + k \bmod q} \bmod p = g^{x_A r + k - Nq} \bmod p = (((g^{x_A})^r \bmod p)((g^k \bmod p) \cdot (g^q \bmod p)^{-N}) \bmod p, \quad (3.1)$$

where

$$N = \lfloor (x_A r + k)/q \rfloor . \quad (3.2)$$

From (3.1), (2.3), (2.2), and (2.5), we get

$$g^{s_p} = (y_A^r \bmod p) \cdot r \cdot 1^{-N} \bmod p = y_A^r r \bmod p.$$

As far as our result is equivalent to the right-hand side (RHS) of (2.8), hence, (2.8) is proved. QED

3.1.2 Proof of (2.14)

We prove that LHS of (2.14) is equal to RHS of (2.14). LHS of (2.14) is equal to RHS of (2.12), hence we prove that RHS of (2.12) equals to RHS of (2.14). We see that they are equal if

$$K_i\delta_i \bmod p \;\; = g^{\hat{s_i}}\, {y_p}^{\hat{e_i}}\delta_i \bmod p$$

Using (2.9) -(2.11) in the LHS, so we have

$$g^{k_i}c(gh)^i \text{mod p} = g^{k_i}g^v h^{-b}\cdot g^i h^i = g^{i+k_i+v}h^{i-b} \text{ mod p}$$

Now, taking RHS, we use (2.9), (2.11), (2.13), and (2.7),

$$g^{\hat{s_i}}\, {y_p}^{\hat{e_i}}\delta_i \bmod p = g^{k_i - s_p\hat{e}_i \bmod q}\,(g^{s_p})^{\hat{e_i}}\cdot(g^v h^{-b}\cdot g^i h^i) \bmod p =$$

$$g^{k_i - s_p\hat{e}_i - Nq + s_p\hat{e}_i + v + i}\cdot h^{i-b} \text{ mod p,}$$

where $N = \lfloor (k_i - s_p\hat{e}_i)/q \rfloor$.

From (2.2), $g^q \bmod p = 1, g^{Nq} \bmod p = 1^N = 1$, and from the last equality,

$$g^{\hat{s_i}}\, {y_p}^{\hat{e_i}}\delta_i \bmod p = g^{k_i+v+i}\cdot h^{i-b} \bmod p$$

Thus, RHS of (2.14) equals to the RHS of (2.12) hence their LHS's are also equal, and (2.14) is proved. QED

3.1.3 Proof of (2.18)

LHS of (2.14) for $i = b$, equals to LHS of (2.18) because of (2.15), Let us prove that the RHS's of (2.14) and (2.18) are also equal:

$$H\left(m_b\,, g^{\hat{s}_b} y_p^{\widehat{e_b}}\delta_b \bmod p\right) = H\left(m_b\,, g^s y_p^e \bmod p\right)$$

$$g^{\hat{s}_b}\, {y_p}^{\widehat{e_b}}\delta_b \bmod p = \; g^s y_p^e \bmod p$$

Using (2.15), (2.16), we have

$$g^s y_p^{\widehat{e_b}} \text{ mod p } = \; g^{\hat{s}_b + v + b \bmod q} y_p^{\widehat{e_b}} \text{ mod p} = \; g^{\hat{s}_b + v + b - Nq} y_p^{\widehat{e_b}} \text{ mod p}\,, \qquad (3.3)$$

where $N = \lfloor (\hat{s}_b + v + b)/q \rfloor$.

From (2.2), since $g^q mod p = 1$, we get $g^{Nq} mod p = 1$.

Substituting our equation into the RHS of (3.3), we get

$$g^s y_p^{\widehat{e_b}} \text{mod p} = g^{\hat{s}_b} g^{v+b} y_p^{\widehat{e_b}} \text{ mod p.}$$

Using (2.9) and (2.11), we have

$$\delta_b = c(gh)^b = g^v h^{-b} (gh)^b = g^{v+b} mod\ p.$$

Thus, it is true, hence,

$$g^s y_p^{\widehat{e_b}} \text{ mod p} = g^{\hat{s}_b} g^{v+b} y_p^{\widehat{e_b}}\ mod\ p = g^{\hat{s}_b} y_p{}^{\widehat{e_b}} \delta_b\ mod\ p\ ,$$

holds, and (2.18) is proved. Q.E.D.

3.2 Adjustment of the Novel Oblivious Signature

There are some issues associated with the signature [19], which we will adjust in this section of the thesis.

In spite of the correctness of (2.8), it can't be verified by B because s/he does not know y_A , hence, we can make a conclusion that in the proxy phase, y_A shall be published, not y_p only as it is specified in the original protocol. Hence, Step 2 of the proxy phase shall be adjusted as follows:

Step 2 Adjusted: The original signer, A, securely sends the pair (r, s_p) to B and y_p , y_A are published.

3.3 Proof of the Voting System Correctness

3.3.1 Proof of (2.34)

Using (2.33) to expand the left hand side of (2.34), we have

$$s_{A,B}^{e_B} = (H(g^{s_A}\ mod\ p)^{d_B}\ mod\ N_B)^{e_B} mod\ N_B = H(y_p') \text{ mod } N_B,$$

due to (2.27), and reminding RSA encryption/decryption, considered in Section 2.3.2. Thus, we get

RHS of (2.34), Q.E.D

3.3.2 Proof of (2.39)

From (2.37),

$$s_B = H(pn, e_B)^{d_B} \bmod N_B .$$

Using the right hand side of (2.39) to prove the equality, and reminding RSA encryption/decryption, we have

$$s_B^{e_B} = (H(pn, e_B)^{d_B} \bmod N_B)^{e_B} \bmod N_B = H(pn, e_B) \bmod N_B ,$$

Q.E.D

3.3.3 Proof of (2.42)

We refer to the proof of (2.14), Section 3.1.2, just that an additional variable pn (pseudo-name) is included in (2.42) and it holds.

3.4 Modification of the Voting System

The modification for the proposed method [19] at the System Setup Phase: $\boldsymbol{y_A}$ shall be published to be used in the proxy phase by B. And this is described in the modified Figure 8, where we have added publishing of $\boldsymbol{y_A}$.

3.5 Summary

In summary, based on the proposed voting [19] protocol, we have proved (2.8), (2.14), (2.18), (2.34), (2.39) and (2.42) not proved in [19]. We have found in the course of proving some problems with the oblivious signature scheme and VS [19] and made necessary adjustments.

Chapter 4

VOTING SYSTEM DESIGN, IMPLEMENTATION AND TESTING

We discussed the Voting System in Chapter 2 and its proof and adjustment in Chapter 3. We proceed in this chapter to the system design, implementation, and testing of the voting system [19]. We design EVS architecture (Section 4.1), specify tools used in its implementation (Section 4.2), and provide information on EVS implementation (Section 4.3) and testing (Section 4.4).

4.1 Voting System Design

Based on the voting system design, we show implementation by Steps 1-6 below for the EVS.

Step 1. Parameters are generated.

Step 2. The original signer, A, delegates authority to B, the proxy signer, and B then publishes the public key to the bulletin.

Step 3. The proxy signer, B, checks whether R, is voter legally registered; if so, a voting certificate is issued to R. B publishes all the certificates to BB (the bulletin board). R checks whether he is registered via BB successfully.

Step 4. The voter, R, makes his choice candidate and receives signature on it from B.

Step 5. The voter, R, casts his vote by sending it to V (the voting center), V immediately publishes a message about the vote cast by R to BB. Every single R can confirm whether his/her vote was received by V; else, voters can resend their votes.

Step 6. When the voting period has ended, B forwards the decrypting key to V, and V verifies and counts the votes. V publishes the result of the votes to BB, where every voter can count and verify all votes.

We design the voting system using context diagram and use-case diagram as shown in Figures 15 and 16, respectively. The context diagram for the EVS represents the entirety of the EVS and how it relates with the actors, it is built for one user, the system administrator, acting as the original signer, A, proxy signer, B, voter, R, and voting center, V.

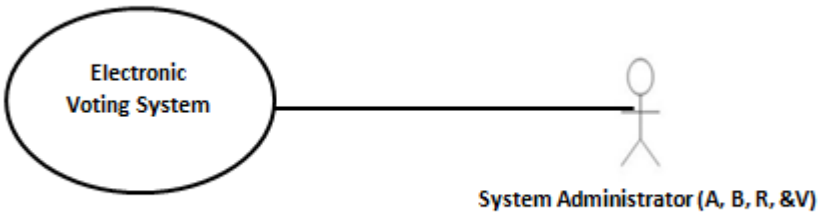


Figure 15: Context diagram for EVS

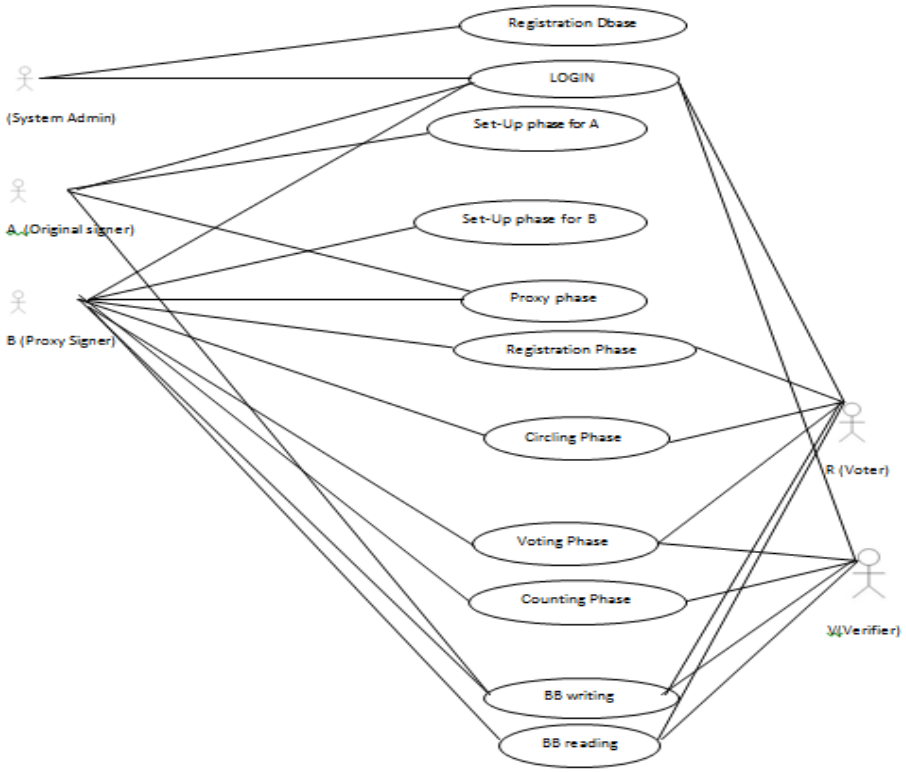


Figure 16: Use-case diagram for EVS

The use-case diagram for EVS in Figure 16 has five actors: System administrator, SA, Original signer, A, Proxy signer, B, Voter, R, and Voting center, V. Based on our EVS implementation, we simulated the work of all these five actors to be done by just one actor, SA. The Registration Dbase use-case is used to populate the database with the information of all Voters. The Login use-case is provided for each actor to be able to have access to the EVS. We have the setup phase use-case for A, and the separate setup use-case for B. We also have the proxy phase for A and B where A delegates authority to B. We also have registration phase use-case, this is where B checks if R is a legal voter by identity number, ID, and registered voters can get Cert(R) and make progress to vote. In circling phase use-case, B checks whether R has voted, if he has, the process gets terminated, else he is given access to continue to vote and make his choice from the candidate list. In voting phase, R encrypts the ballot and sends to V in order to hide his choice until the end of the voting period.

From the counting phase use-case, every candidate selection tallies only with corresponding pseudo-name (pn) available from BB.

BB writing use-case provides access for an actor to write on the bulletin board so that everyone could have access to vital information that is specific to them, voters from it could ascertain whether their names and votes have been received by the vote center and election results can be verified. While, for the BB reading use-case is available for everyone to be able to verify their information on the screen; you could also click on the BB button on the menu page to have access to the page, according to Figure 22.

4.2 Tools Used for EVS Implementation

We implemented the EVS [19] using Windev express version 17 [37, 38] which has a built-in database called Hyper File SQL. WinDev is an integrated platform for software development. Most software developed with WinDev offers a set of advanced features, you can download it from [37, 38], and the installation steps and work environment is shown below, Figures 17-20.

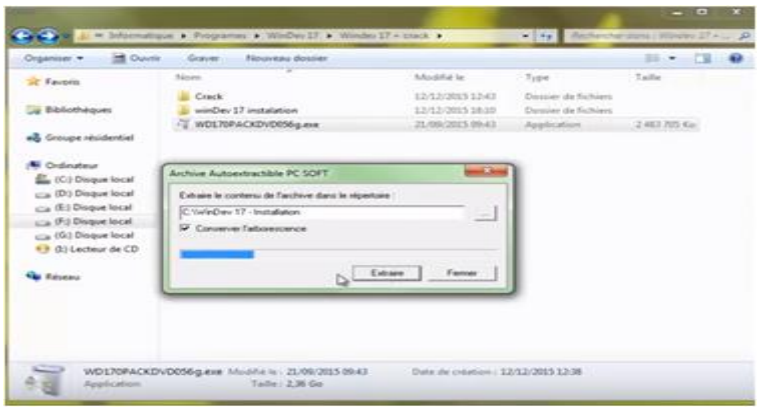


Figure 17: WD Full Application Install.exe

To install WinDev17, click on the WD full Application install.exe icon, as shown in Figure 17:

1. Click on “installation de Windev 17” during installation, as shown in Figure 18.

Figure 18: Installation Phase for French Language

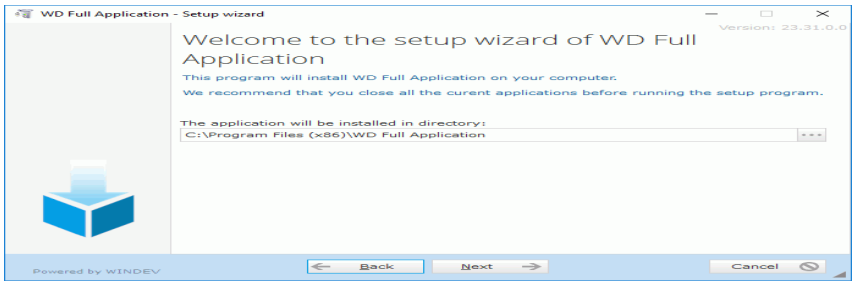


Figure 19: Setup Directory for Windev Application

2. Validate the setup directory by clicking on the application, as shown in Figure 19.

3. Move to next step and end the setup for the application.

4. Then the application starts up. Click to validate the steps of the setup.

5. Then we have the work environment, as shown in Figure 20.

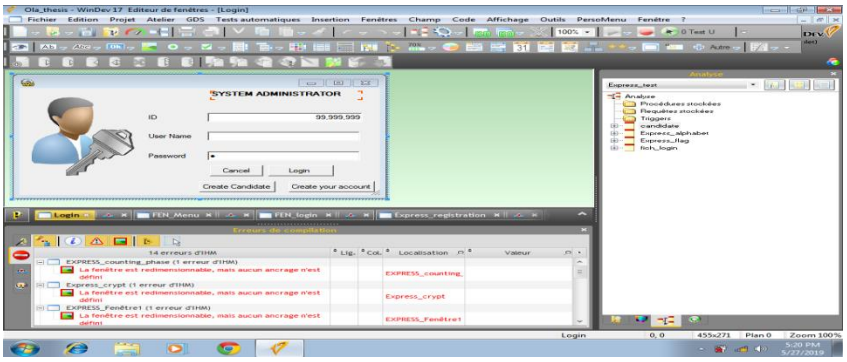


Figure 20: Windev Work Environment

More details on how to install Windev Application are given in the webpage in [39].

4.3. Voting System Implementation

We model the EVS as proposed by [19] based on the context diagram in Figure 15 and use-case diagram in Figure 16. It will be run by a single user, SA, representing all actors. In the Login use-case, SA logs on and then moves to the Menu page use-case to select whatever process he wants to choose either A, B, R, or V. Figure 21 is the screenshot for the Login use-case and Figure 22 is the screenshot for the Menu page.

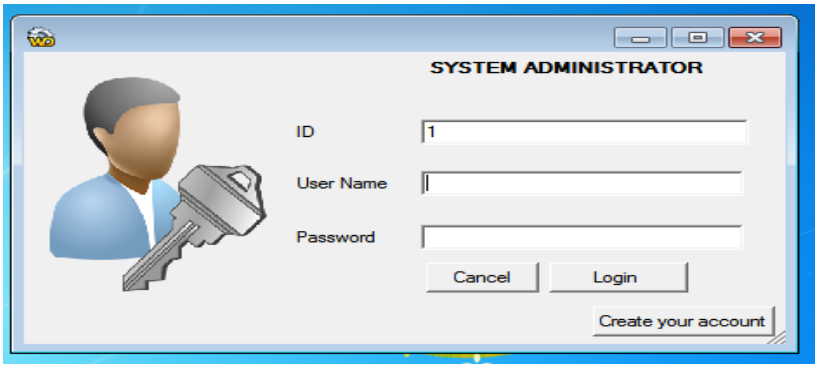


Figure 21: Login screenshot

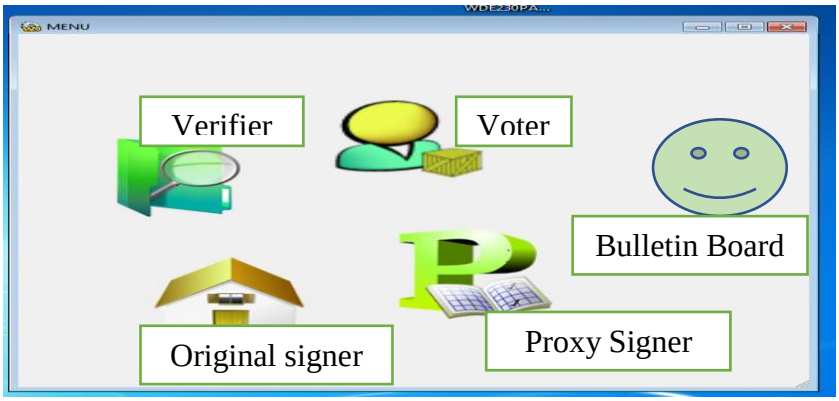


Figure 22: Menu page screenshot

4.3.1. Implementation of Setup Phase for A and B Use-Case

Figure 23 shows the screenshot of the system setup phase use-case for both A and B, Figure 16 described the use-case for the setup phase for A and B in Section 2.4.4.1 of our protocol design. In Step 1, A selects two large prime values, p and q (see Appendices A.1.2, lines #9-28, and A.1.3, lines #29-57), according to (2.1). In Step 2, two generators of order q, which are h and g according to (2.2) (Appendix A.1.4, lines #59-66, and Appendix A.1.5, lines #68-80) are selected by A. In Step 3, A chooses $x_A \in Z_q^*$ (Appendix A.1.12, lines #156-160), and calculates y_A, defined by (2.3) (Appendix A.1.6, lines #81-83). Then, A publishes $p, q, g, h,$ and y_A values on BB (Appendix A.1.1, lines #1-4) as described on Figure 16 as BB writing use-case. In Step 4, B enters x_B and computes y_B, according to (2.4) (Appendix A.1.9, lines # 128-131). In Step 5, B chooses two large prime numbers p_B and q_B (Appendices A.1.7, A.1.8, lines #84-127). In Step 6, the product of p_B and q_B is stored in N_B according to (2.24), (Appendix A.1.8, lines#116-117). In Step 7, Euler totient function, $\emptyset(N_B)$, is calculated according to (2.25) (Appendix A.1.8, lines#119-121). In Step 8, B selects RSA public key, e_B, meeting condition (2.26). In Step 9, B calculates private key d_B, according to (2.27) (Appendix A.1.10, lines #131-151). Then B publishes N_B, e_B, y_B, on BB (Appendix A.1.11, lines #152-155).

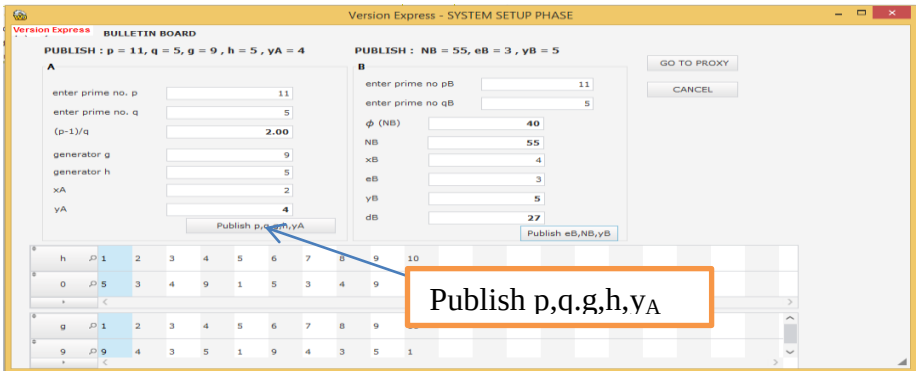


Figure 23: System Setup Phase for A and B

4.3.2 Implementation of Proxy Phase Use-Case

Figure 24 shows the screenshot of the proxy phase, Figure 16 described the use-case for the proxy phase described in Section 2.4.4.2 of our protocol design. In this phase, communication is made between A and B. In Step 1, A chooses a random number, k, and computes r_A, s_A and y_p' according to (2.28)-(2.30). (Appendix A.2.2, lines# 165-168) In Step 2, A encrypts (r_A, s_A) using RSA, and it is forwarded to B, (Appendix A.2.2, lines # 164-180). In Step 3, B decrypts the message (r_A, s_A) using RSA private key (d_B, N_B) (Appendix A.2.5, lines #231-239) and checks condition (2.31) to know if values have been compromised, (Appendix A.2.3, lines #181-212). If (2.31) is true, B accepts the message and computes s_p as his own secret key signature, according to (2.32). In Step 4, B generates $S_{A,B}$ signature according to (2.33) and forwards it to A by clicking the send button (Appendix A.2.4, lines #213-230). In Step 5, A checks if the signature $S_{A,B}$ is valid according to (2.34) (Appendix A.2.6, lines #240-249). If (2.34) is true, the original signer A publishes y_p' to BB (Appendix A.2.1, line #156-158).

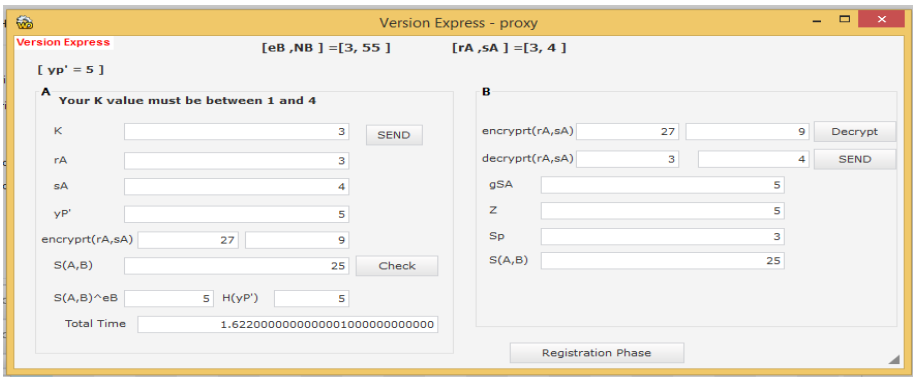


Figure 24: Proxy Phase

4.3.3 Implementation of Register Phase Use-Case

Figure 25 shows the screenshot of the register phase, Figure 16 described the use-case for the register phase described in Section 2.4.4.3 of our protocol design. Every eligible voter name exists already in the database called IDExpress_Flag designed in HyperFileSQLWindev; this is detailed in Section 4.3.7. Figure 25 shows that Voters R records are already populated in the database on a grid. It is populated by A with R records, so R updates records of password (pw) and pseudo-name (pw), according to Section 2.4.4.3. In Step 1, R encrypts his id, pseudo-name and hash of his password, according to (2.35) (Appendix A.3.1, lines #250-276, Appendix A.3.2, lines #277-301, and Appendix A.3.3, lines #302-325), using (N_B, e_B) published on BB and sends it to B, Figure 11. In Step 2, B decrypts it using d_B and checks if the ID is valid, sets the vote flag to zero meaning the voter has not voted, and then allows the voters The proxy signer B computes key_R and s_B, according to (2.36), (2.37) respectively; (Appendix A. 3.4, lines #326-346) and then forwards the certificate of voter R (Appendix A.3.5). R sees it, and proceeds to vote (Appendix A.3.6, lines# 360-388).

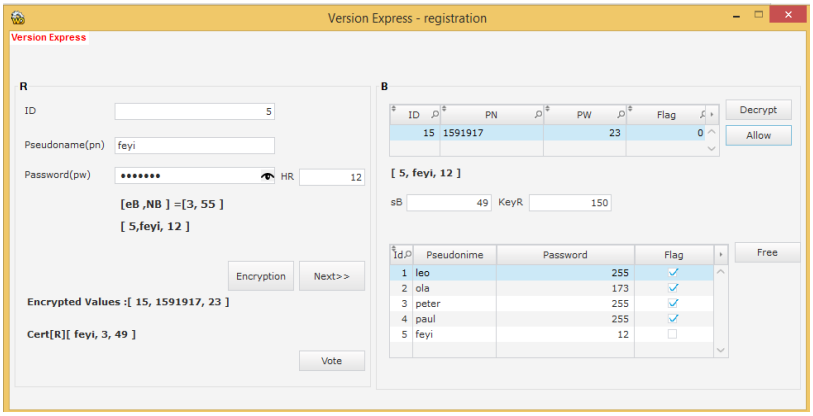


Figure 25: Registration phase

4.3.4 Implementation of Circling Phase Use-Case

Figure 26 shows the screenshot of circling phase, Figure 16 described the use-case for the circling phase described in Section 2.4.4.4 of our protocol design. In Step 1, B sends a random number r after the request from R, (Appendix A.4.1, lines # 389-392). In Step 2, R proceeds to calculate H'_R using an hash function, according to (2.40), (Appendix A.4.2, lines# 393- 404), this is where the use of the oblivious signature comes in, R picks a random number v used as a blinding factor (Appendix 4.3, lines# 405- 408), and computes c for all candidates and picks b as the candidate of his or her choice, m_b, from the whole candidates list $(m_i), i = 1,..,n$, according to (2.9), (Appendix 4.4, lines# 409-429), and forwards the values to B, (Appendix 4.5, lines# 430-457). In Step 3, B checks whether the hash function are equal, according to (2.41), (Appendix 4.6, lines# 458-469), then B chooses some k_i and calculates value for $K_i, \delta_i, \widehat{e_i}$, and $\widehat{s_i}$ for all i, (Appendix A.4.7, lines #470-501), B sets the flag(pn) to 1 and sends $\widehat{e_i}$, and $\widehat{s_i}$ for all i back to R, according to (2.10), (2.11), (2.12), (2.13), (Appendix A.4.8, lines #502-516). In Step 4, R, calculates y_p, according to (2.19), (Appendix A.4.4, lines #409-414), for all i, he calculates for δ_i, and checks if $\widehat{e_i}$, according to (2.11), (2.42) respectively, is correct, (Appendix

A.4.8, lines #517-522), then R computes s and e, according to (2,15), (2.16) respectively as the final signature (Appendix A.4.9, lines #527-545).

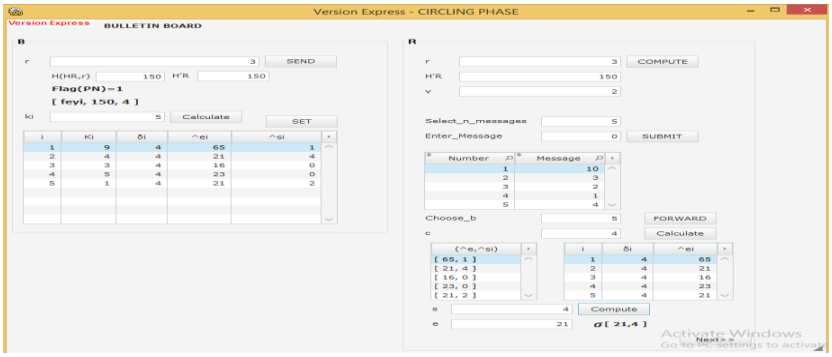


Figure 26: Circling Phase

4.3.5 Implementation of Voting Phase Use-Case

Figure 27 shows the screenshot of voting phase, Figure 16 described the use-case for the voting phase described in section 2.4.4.5 of our protocol design. In Step 1, R calculates $H(H(pw), e_B)$, (Appendix A.5.1, lines #546-555) and uses it as a symmetric key to encrypt $(m_b, \sigma(m_b))$, produces a cipher C_R, (Appendix A.5.2, lines #556-580), then send the certificate to the voting center V, (according to Appendix A.5.3, lines #581-588). In Step 2, V examines it, according to (2.39), (Appendix A.5.4, lines # 589-619) and sends it to BB; (Appendix A.5.5, lines #620-632). In Step 3, every voter can check whether his or her ballot is received by V via BB, else the voter resends his ballot.

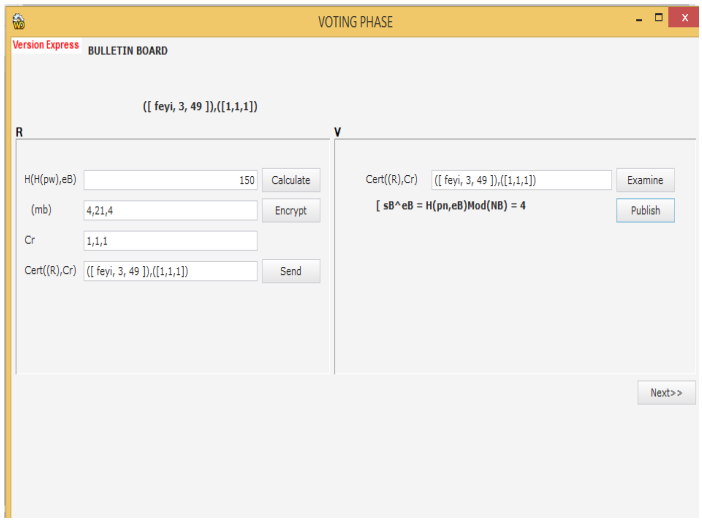


Figure 27: Voting Phase

4.3.6 Implementation of Counting Phase Use-Case

Figure 28 shows the screenshot of the counting phase, Figure 16 described the use-case for the counting phase described in section 2.4.4.6 of our protocol design; this phase is done between B and V. In Step 1, the key generated by B during the registration phase is sent to V; according to (2.44), (Appendix A.6.1, lines #633-636). In Step 2, V decrypts C_r, (Appendix A.6.2, lines #637-646) and V now publishes $(\text{Cert(R)}, C_R, m_b, key_R)$ on BB, (Appendix A.6.3, lines #647-655, and V calculate y_p, according to (2.19), (Appendix A.6.4, lines #656-663), then, V verifies e, according to (2.45), (Appendix A.6.5, lines #664-687). In Step 3, V publishes the election result on BB, (Appendix A.6.6, lines #688-700) and everyone can verify and count the ballots via BB.

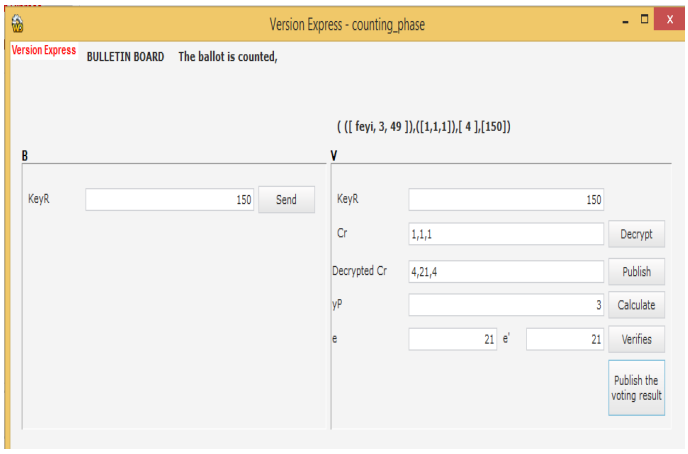


Figure 28: Counting Phase

4.3.7 Database Implementation

There are several ways available to connect to database. The connection to database in this thesis is done directly by connecting to a HFSQL Classic database. The main operation that is performed in order to make use of WDSQL is that we have to establish foremost connection to database. Once this connection is done, we can now have access to run and create SQL queries on the used database.

The database used is inbuilt with Windev application called HFSQL, as shown in Figure 29 for the overall view of the database; we have a database with four tables. For the Registration of voters and voter's choice, Express_flag table was used, as shown in Figure 30. For login information for all users, Fich_Login table is used to store records for each user that have access to the EVS, as shown in Figure 31. For character coding table, we used the Express_alphabeth table to store the values, as shown in Figure 32. For the database of contestant and vote result we used candidate.fich as shown in Figure 33.

Figure 29: Database view [Overall]

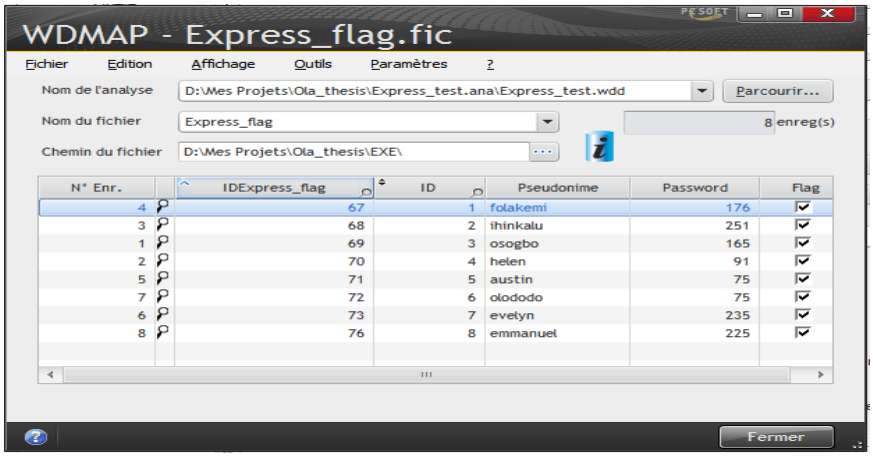


Figure 30: Table for Express_flag

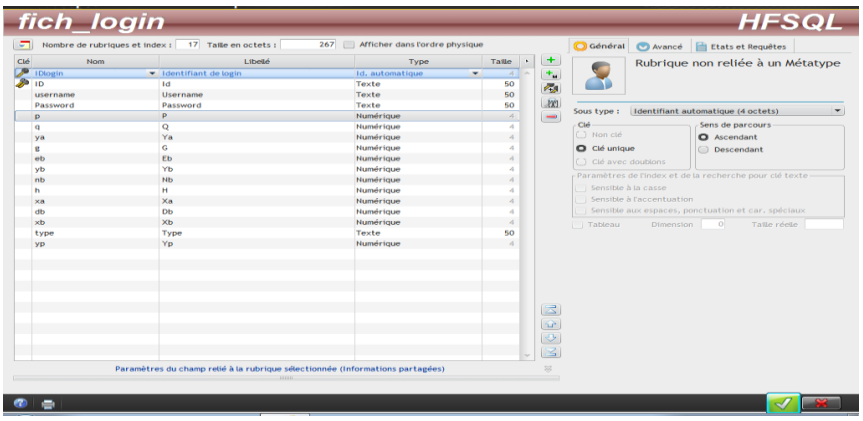


Figure 31: Table for Fich_login

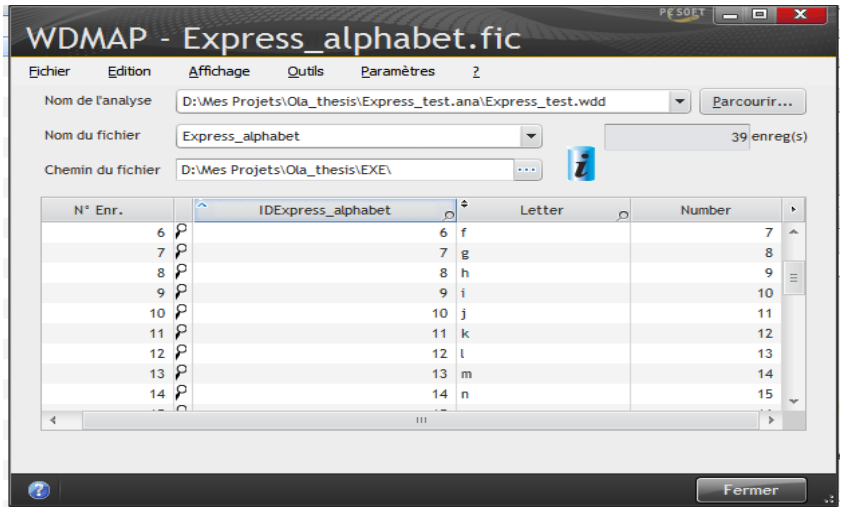


Figure 32: Table for Express alphabeth

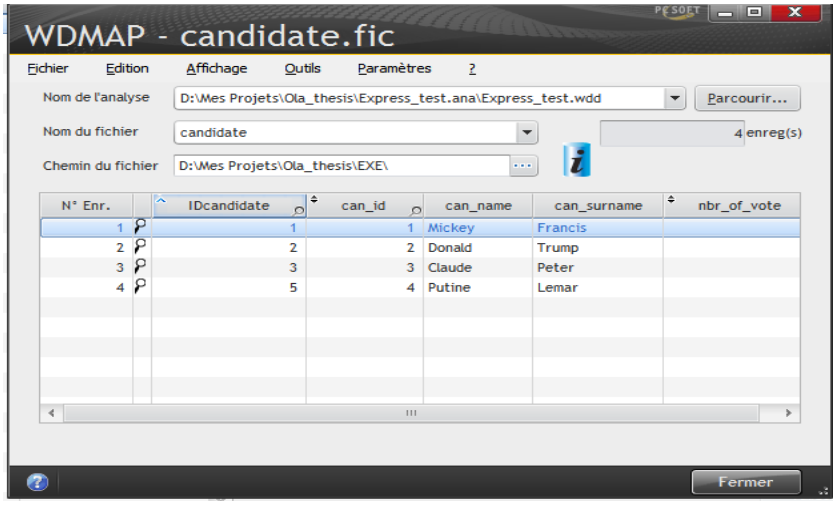


Figure 33: Contestant and results table

4.4 Voting System Testing

In software development, the testing phase is of uttermost importance. After the software was developed the voting protocol was also tested, so to ascertain that it met the necessary requirements as proposed by [19] and also the modifications made in these thesis. Below are the snapshots for each of the phases of the voting protocol.

4.4.1. Testing of System Setup Phase Use-Case

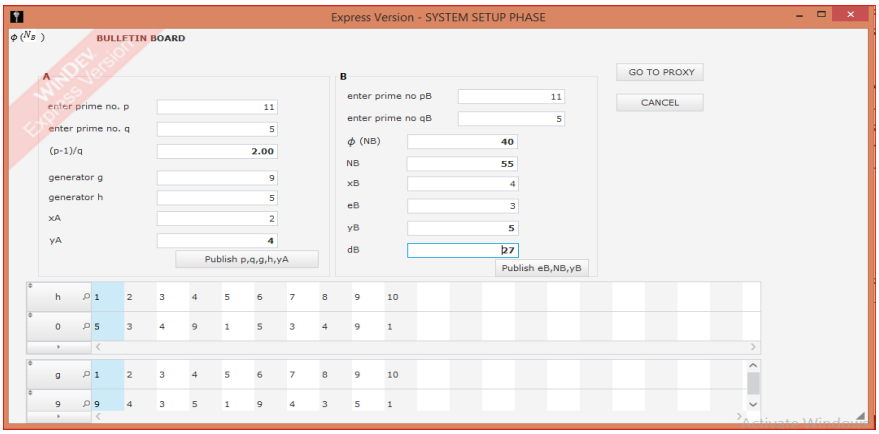


Figure 34: Testing for ystem setup phase

Tested result for Figure 34 has been explained in example1 and example 2 of section 2.4, for Figure 34 the Right hand side is original signer A segment and the left hand side is the proxy signer B segment.

According to (2.1). Let $p = 11, q = 5$ be our prime numbers chosen, $p - 1 = 11 - 1 = 10, 10/5 = 2$.

From equation (2.2), $h = 5$, $g = 9, Ord\ 5 = q$ $and\ Ord\ 9 = q = 5$.

From equation (2.3), $p = 11, Z_q^* = \{1, 2, 3, 4\}$.
Let $x_A = 2$, and $x_B = 4$ then $y_A = g^{x_A} \bmod 11, = 9^2\ mod\ 11 = 4$.

From equation (2.4), we have $y_B = 9^4\ mod\ 11 = 5$.

Let $p_B = 11$, $q_B = 5$.

According to (2.24), $N_B = 11 \cdot 5 = 55$.

According to (2.25), $\emptyset\ (N_B) = (11 - 1)(5 - 1) = 40$.

According to (2.26), $e_B = 3$. According to (2.27), d_B =27.

Now, we need to compute $d_B = e_B^{-1} \bmod \emptyset(N_B)$ by using backward substitution of GCD algorithm:

According to GCD:

$$40 = 3 \cdot 13 + 1$$

$$13 = 1 \cdot 13 + 0$$

Therefore, we have:

$$1 = 40 - 3 \cdot 13$$

Hence, we get $d_B = e_B^{-1} \bmod \emptyset(N_B) = e_B^{-1} \bmod 40 = -13 \; mod \; 40 =$

$(27 - 40) \; mod \; 40 = 27$

So, the public key is {3, 55} and the private key is {27, 55}.

4.4.2. Testing of Proxy Phase Use-Case

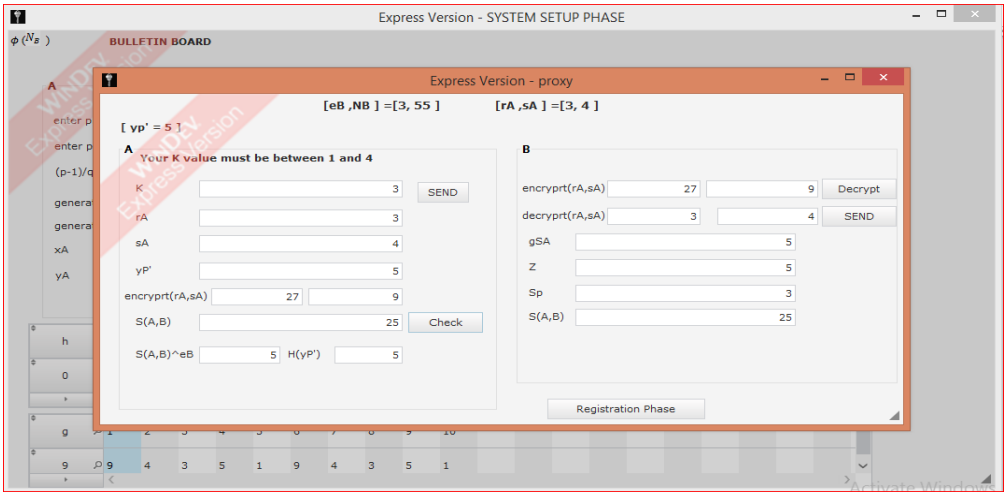


Figure 35: Testing for proxy phase

Tested result for Figure 35 has been explained in Example 3 of section 2.4, for Figure 35 the Right hand side is original signer A segment and the left hand side is the proxy signer B segment.

Recall (2.5) r was used as our notation, we will now use r_A in place of r for this example

$r_A = 9^3 \bmod 11 = 3$

For (2.20),

$$s_A = 2 \cdot 3 + 3 \; mod \; 5 = 4$$

$$(r_A, s_A) = (3,4)$$

$$y'_p = g^{s_A} \, mod \, p = \, 9^4 \, mod \, 11 = 5$$

the original signer, A, encrypted pair $(r_A, \ s_A)$ $(= (3,9))$ using (e_B, N_B) $(= (3, 55))$, and $y'_p \ = 5$ will be forwarded to B. The proxy signer B must decrypt the message using (r_A, s_A) $(= \ (3,9))$ using $(d_B, N_B) = (27,55)$

(For our numbers. , $5 = g^{s_A} = r y_A^r = 3 \cdot 4^3 = 5 mod \ 11 = 5$)

which holds for our values. The proxy signer B accepts computes s_p

From (2.2) –(2.6), which hold, $s_p = 9 + 4 \, mod \, 5 = 13$

From (2.33),

$$S_{A,B} = H(9^9 \, mod \, 11 = 5)^{27} \, mod \, 55 = \text{H}(5)^{27} \text{ mod } 55 = 53^{27} \, mod \, 55 = 37,$$

$$s_{A,B}^{e_B} = 37^3 mod \, 55 = H \, (5) \ = \ 53, \ \text{(Note: The ASCII value for 5 is 53)}$$

$$H \, (y'_p) \, mod N_B = H(5) mod \, 55 = H(5) = 53.$$

4.4.3 Testing of Register Phase Use-Case

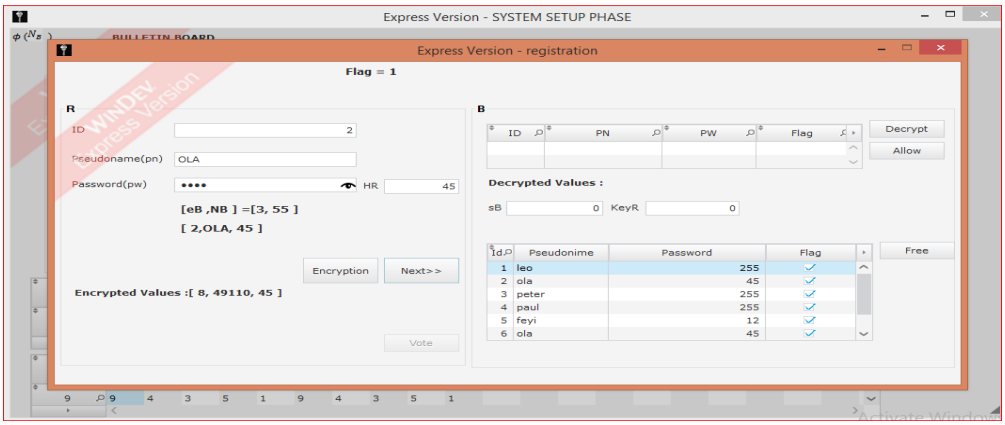


Figure 36: Testing for Registration Phase

Tested result for Figure 36 has been explained in Example 4 of section 2.4, for Figure 36 the Right hand side is the voter R segment and the left hand side is the proxy signer B segment.

From (2.39), R enters Password as “FEYI” and pseudo-name as “OLA”

Each of the ASCII value for each letter is taken added up modulo N

H_R = H(pw) =70+69+89+73 = $301 \bmod 55 = 26$

Encryption is done for each of id, pseudo-name and password, i.e., (id, pn, H(pw))

(e.g.,=(2,OLA,26)) using (e_B, N_B)= (3, 55)) which when encrypted, for the pseudo-name (pn), we introduced Character Coding Table, illustrated in Table 3

We now encrypt the value of each character. For OLA: 16-13-2, this takes each value before the dash and encrypt it as follows: we used $(e_B, N_B) = (3, 55))$ as our encryption keys, we have $16^3 mod\ 55 = 26, 13^3 mod\ 55 = 52, 2^3 mod\ 55 = 8$ after these encryption, we now concatenate each encrypted the value to get $26||52||8$ taken out the dash(“-”) to form "26528". We have $(8, 26528, 31)$ after the encryption for each of them.

The encrypted $(id, pn, (pw))$ is decrypted by B to give us $(2, OLA, 26)$ then R is checked to be a legal voter and the flag (pn) value is set to be zero,

From (2.36), calculated to give (e.g. $Key_R = H\ (45,3) = 156 = 52 + 53 +$

$51 \bmod 55 = 46$

From (2.37) , $s_B = H(OLA, 3)^{27} \bmod 55 = (H(OLA||3) = 79 + 76 + 65 + 51 =$

$271 \bmod 55 = 51 = s_B = (51)^{27} \bmod 55 = 6$

This is forwarded to R as certificate, $Cert(R) = (pn, e_B, s_B) = (OLA, 3,6)$

The certificate, $Cert(R) = (pn, e_B, s_B) = (OLA, 3,6)$, is forwarded to R.

Verification is done by voter, R, according to (2.39):

$s_B^{e_B} = 6^3 \bmod 55 = 51$

H (OLA, 3) mod $55 = 79 + 76 + 65 + 51 = 271 \bmod 55 = 51$, and it is verified because (2.39) holds.

4.4.4 Testing of Circling Phase Use-Case

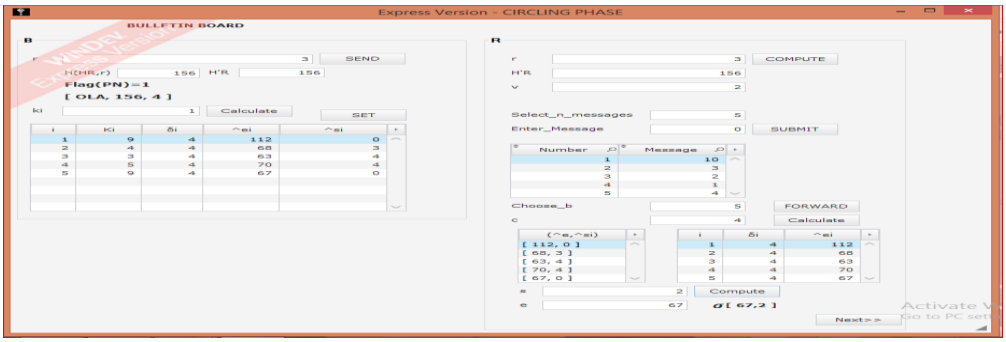


Figure 37: Testing for Circling Phase

Tested result for Figure 37 has been explained in Example 5 of section 2.4, for Figure 36 the Right hand side is the proxy signer B segment and the left hand side is the voter R segment.

The proxy signer B sends a random chosen number $r = 3 \in_R Z_q^* = \{1,2,3,4\}$ to the voter R after which the voter R is requested to login.

The voter R does some computations from (2.44)

$H_R' = H(H(pw), r) = H(45,3) = 156 mod\ 55 = 46$

Also, from (2.9), for our values, $n = 6, \{m_1, m_2, \ldots, m_n\} = \{10, 3, 2, 1, 4, 7\}, b = 5,$

$v = 2, c = g^v h^{-b} \bmod p = 9^2\ 5^{-5} \bmod 11 = 9^2 \cdot 3125^{-1} \bmod 11 =$

$3125^{-1} \bmod 11 = 1,$ this can be rewritten as, $3125 \cdot x = 1 \bmod 1, then\ x = 1,$

$=9^2 \cdot 1 \bmod 11 = 4$

$(m_b \in \{m_1, m_2, \ldots, m_n\})$, he then forwards the message $(pn, H_R', c) = (OLA, 46,4)$ to the proxy signer B

For (2.41) $\mathrm{H}'_{\mathrm{R}} = H\,(H_R, r) = H(45,3) = 156 mod 55 = 46$

From (2.10)

$k_i = \{1,2,3,4,1,2\}$

$k_1 = 1,\ k_2 = 2,\ k_3 = 3,\ k_4 = 4,\ k_5 = 1,\ k_6 = 2$

$K_1 = 9^1\ mod\ 11 = 9, K_2 = 9^2\ mod\ 11 = 4, K_3 = 9^3\ mod\ 11 = 3, K_4 =$

$9^4\ mod\ 11 = 5$

$K_5 = 9^1\ mod\ 11 = 9, K_6 = 9^2\ mod\ 11 = 4$

From (2.11)

$\delta_1 = 4(9 \cdot 5)^1 mod\ 11 = 4, \delta_2 = 4(9 \cdot 5)^2 mod\ 11 = 4, \delta_3 = 4(9 \cdot$

$5)^3 mod\ 11 = 4.$

$\delta_4 = 4(9 \cdot 5)^4 mod\ 11 = 4, \delta_5 = 4(9 \cdot 5)^5 mod\ 11 = 4, \delta_6 = 4(9 \cdot$

$5)^6 mod\ 11 = 4$

From (2.15) it used two variables for the hash function, but here we are introducing for three variables for this hash function.

Using a hash function that has three inputs introduced in (2.14).

$$\widehat{e_i} = H(m_i\ , pn, K_i\delta_i\ mod\ p\ ), \qquad (2.43)$$

$\widehat{e_1} = H(10, OLA\ ,9 \cdot 4\ mod\ 11\ ) = H(10, OLA, 3) = 49 + 48 + 79 + 76 + 65 +$

$51\ mod\ 55 = 38$

$\widehat{e_2} = H(3, OLA\ ,4 \cdot 4\ mod\ 11\ ) = H(3, OLA, 5) = (51 + 79 + 76 + 65 +$

$53) mod\ 55 = 49$

$\widehat{e_3} = H(2, OLA\ ,3 \cdot 4\ mod\ 11\ ) = H(2, OLA, 1) = 50 + 79 + 76 + 65 +$

$49\ mod 55 = 44$

$\widehat{e_4} = H(1, OLA\ ,5 \cdot 4\ mod\ 11\ ) = H(1, OLA, 9) = 49 + 79 + 76 + 65 +$

$57 mod\ 55 = 51$

$\widehat{e_5} = H(4, OLA, 9 \cdot 4 \, mod \, 11) = H(4, OLA, 3) = 52 + 79 + 76 + 65 +$

$51 \, mod 55 = 48$

$\widehat{e_6} = H(7, OLA, 4 \cdot 4 \, mod \, 11) = H(7, OLA, 5) = 55 + 79 + 76 + 65 +$

$53 \, mod \, 55 = 53$

From (2.13), recalling (2.2), (2.10),

$$\widehat{s_1} = 1 - 3 \cdot 38 \bmod 5 = 2,$$

$$\widehat{s_2} = 2 - 3 \cdot 49 \bmod 5 = 0,$$

$$\widehat{s_3} = 3 - 3 \cdot 44 \, mod \, 5 = 1,$$

$$\widehat{s_4} = 4 - 3 \cdot 51 \, mod \, 5 = 1,$$

$$\widehat{s_5} = 1 - 3 \cdot 48 \, mod \, 5 = 2,$$

$$\widehat{s_6} = 2 - 3 \cdot 53 \, mod \, 5 = 3$$

. $$(\hat{e}_i, \widehat{s_i}) = (38,2), (49,0), (44,1), (51,1), (48,2), (53,3)$$

$$s = 2 + 2 + 5 \, mod \, 5 = 4$$

according to (2.17), $\sigma(m_5) =$ (e,s) =(48,4).

4.4.5. Testing of Voting Phase Use-Case

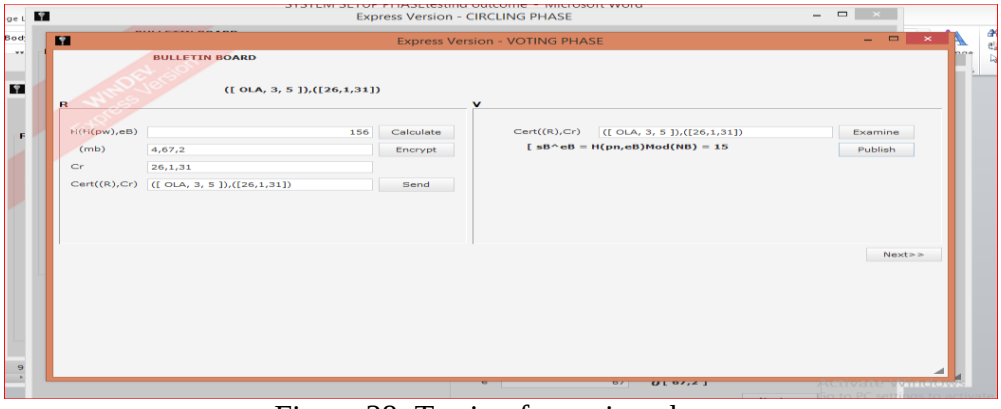


Figure 38: Testing for voting phase

Tested result for Figure 38 has been explained in Example 6 of Section 2.4, for Figure 38 the Right hand side is the voter R segment and the left hand side is the verifier V segment.

R uses a symmetric key $key_R = H(45,3) = 46$ to encrypt $(m_b, \sigma(m_b))$ =([4], [48,4]), using Caesar cipher encryption; $c = p + k \bmod N$, where c is cipher, p is plaintext, and k is the key (the shift), therefore, we have $4 + 46\ mod\ 55 = 50$, $48 + 46\ mod 55 = 39$, 4+46 mod55=50; which produces $C_R = (50,39,50)$. It will now send $(Cert(R), C_R)$ as $([\ OLA, 3, 5\ ]), ([50,39,50])$ to the voting center, V.

The voting center, V checks if

$s_B^{e_B} = 6^3 \bmod 55 = 51 = H\ (OLA, 3) \bmod\ 55 = 79 + 76 + 65 + 51\ mod 55 = 51$

is true, V publishes (Cert(R), C_R) on BB and each voter can check its presence there.

4.4.6 Testing of Counting Phase Use-Case

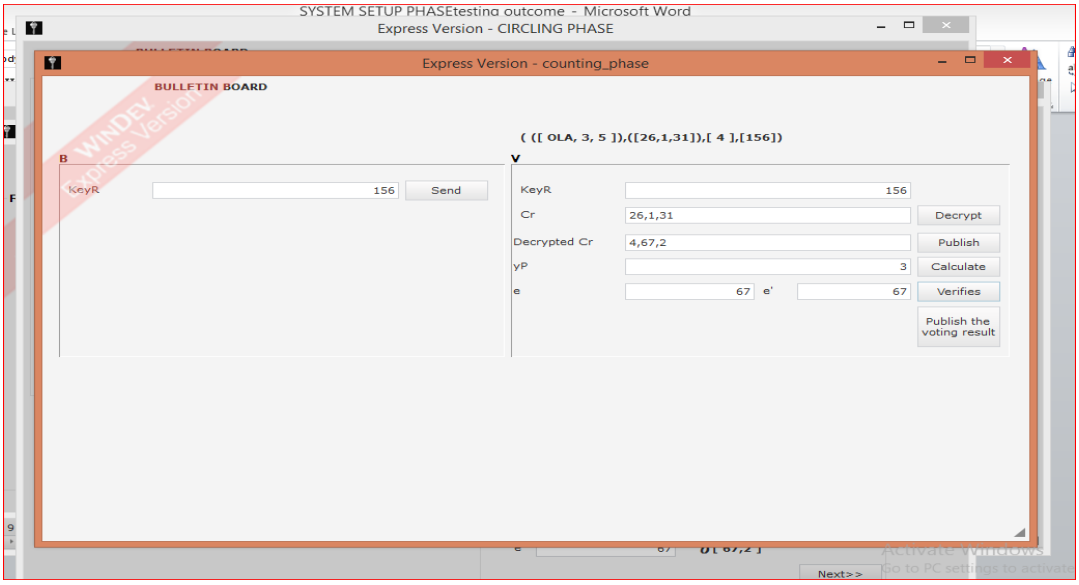


Figure 39: Testing for Counting Phase

Tested result for Figure 39 has been explained in Example 7 of Section 2.4, for Figure 39 the Right hand side is the proxy signer B panel and the left hand side is the verifier V panel.

The proxy signer, B, forwards the key, $key_R = \mathrm{H}(45,3) = 46$, to V. The voting center decrypts using symmetric encryption, $\mathrm{C_R} = ([50,39,50])$ using $key_R = 46$, with formula p = c - k mod N, where c is cipher, p is plaintext and k is key, therefore we have 50-46=4, 39-46=-7+55=48, 50-46=4; we have this result ([4], [48,4]), then it publishes $(Cert(R), C_R, m_b, key_R) = ((OLA, 3,5)$, (4,48,4),4,46),

From (2.18), also adding a final variable

$$y_p = y_p' y_B \mathrm{mod\ p} = 5 \cdot 5 \bmod 11 = 3$$

$48 = \mathrm{e} = \mathrm{H}(m_b, pn, g^s y_p^e \mathrm{mod}\ p) = \mathrm{H}(4, \mathrm{OLA}, 9^4 \cdot 3^{48} mod\ 11) = H(4, OLA, 3) =$

$(52 + 79 + 76 + 65 + 51) mod\ 55 = 48$, which is correct.

4.5. Summary

We designed the EVS architecture in section 4.1, we also specify the tools used for the implementation of our EVS in section 4.2 as Windev 23 Express Version, and we also provided screen shot of our implementation with respect to various design phase in Section 4.3 and finally we obtain results for our design and implementation to produce the screen shots for our testing in Section 4.4.

Chapter 5

COMPARISON OF COMPUTATION TIME FOR KNOWN AND IMPLEMENTED VOTING SYSTEMS

This chapter shows a comparison of the implemented and tested voting protocol as against the known one [19] for oblivious and proxy signature.

For the known Voting protocol [19], it was not specified the software android mobile package that was used by for the implementation of our voting protocol, but the computation time in milliseconds gave the results [19] in Table 4, as shown in Section 2.4.4.7, where PP stands for the Proxy Phase, RP stands for the Registration Phase, CiP stands for the Circling Phase, VP stands for the Voting Phase and CoP stands for the Counting Phase. Other notation like A is the original signer, B is the proxy signer, R is the voter, and V is the verifier. After we used Windev 23 express edition for our implementation and testing, the result obtained is fair and better than that of the known [19] computation time. There is an inbuilt function in Windev, it is a function that is used to calculated runtime in seconds. To get the mathematical calculation of the time it takes for a process to complete its operation, we use an inbuilt function in Windev 23 Edition to perform the computation time. We call the function of the clock called ChronoDebut() at the start and at the end (ChronoFin) using the subroutine code which measures time in seconds, and then we further divide by 1000 to get processing time in millisecond. Appendix A, Source code lines (# 177,181, 195,204,205,206). These average computation time was tested 4 times

(according to Appendix B, Experimental Results screen shots) and the average of the results was computed.

These functions are used to calculate the time passed between the start (ChronoDebut) and the end (ChronoFin) , we used this to calculate the average computation time from one phase to another. Table 5 shows the average computation time measured in milliseconds for the known [19] and the implemented.

Table 5: Average computation time for EVS (milliseconds) for 6 phases

Phase	SP	PP		RP		CiP		VP		CoP
Role	A | B	A	B	R	B	R	B	R	V	V
Time[19]	-	42.4	31.5	22.9	19.6	31.2	31	19.8	10.4	20.3
Time (Implemented)	-	4.2	3.1	2.6	2.1	3.2	3.5	2.0	1.4	2.0
Relative change (%)	-	90	90.2	88.6	89.3	89.7	88.7	89.8	86.5	90.1

We shall do the comparison based on each of the six phases of the implementation using Relative Change [42] to compare with the known system [19].

The formula for Relative change is represented as

$$\text{Relative Change}\left(\text{x}, x_{reference}\right) = \frac{actual\ change}{|x_{reference}|} \cdot 100\%$$

$$= \frac{|x - x_{reference}|}{|x_{reference}|} \cdot 100\%$$

where x is time in (milliseconds) for the implemented EVS for each phase, and $\boldsymbol{x_{reference}}$ is the time in (milliseconds) for the known EVS for each phase. So after computation, the results obtained for all the phases are:

Phase (PP): For **A** in known [19], it took 42. 4ms to complete the process while for implemented, an average of 4.191ms

$RC(4.2, 42.4) = \frac{|4.2-42.4|}{42.4} \cdot 100\% = \frac{38.2}{42.4} \cdot 100\% = 0.90 \cdot 100\% = 90\%$

For **B** in known [19], it took 31.5ms to complete the process while for implemented, an average of 3.1ms

$RC(3.1, 31.5) = \frac{|3.1-31.5|}{31.5} \cdot 100\% = \frac{28.4}{31.5} \cdot 100\% = 0.903 \cdot 100\% = 90.2\%.$

Phase (RP): For **R** in known [19], it took 22.9ms to complete the process while for implemented, an average of 2.6ms

$RC(2.6, 22.9) = \frac{|2.6-22.9|}{22.9} \cdot 100\% = \frac{20.3}{22.9} \cdot 100\% = 0.886 \cdot 100\% = 88.6\%$

For **B** in known [19], it took 19.6ms to complete the process while for implemented, an average of 2.1ms

$RC(2.1, 19.6) = \frac{|2.1-19.6|}{19.6} \cdot 100\% = \frac{17.5}{19.6} \cdot 100\% = 0.893 \cdot 100\% = 89.3\%.$

Phase (CiP): For **R** in known [19], it took 31.2ms to complete the process while for implemented, an average of 3.2ms

$RC(3.2, 31.2) = \frac{|3.2-31.2|}{31.2} \cdot 100\% = \frac{28}{31.2} \cdot 100\% = 0.897 \cdot 100\% = 89.7\%$

For **B** in known [19], it took 31ms to complete the process while for implemented, an average of 3.5ms

$RC(3.5, 31) = \frac{|3.5-31|}{31} \cdot 100\% = \frac{27.5}{31} \cdot 100\% = 0.887 \cdot 100\% = 88.7\%.$

Phase (VP): For **R** in known [19], it took 19.8ms to complete the process while for implemented, an average of 2ms

$RC(2, 19.8) = \frac{|2-19.8|}{19.8} \cdot 100\% = \frac{17.8}{19.8} \cdot 100\% = 0.898 \cdot 100\% = 89.8\%$

For **V** in known [19], it took 10.4ms to complete the process while for implemented, an average of 1.4ms

$RC(1.4, 10.4) = \frac{|1.4-10.4|}{10.4} \cdot 100\% = \frac{9}{10.4} \cdot 100\% = 0.865 \cdot 100\% = 86.5\%.$

Phase (CoP): For **V** in known [19], it took 20.3ms to complete the process while for implemented, an average of 2ms

$RC(2, 20.3) = \frac{|2-20.3|}{20.3} \cdot 100\% = \frac{18.3}{20.3} \cdot 100\% = 0.901 \cdot 100\% = 90.1\%.$

We conclude that the value for the implemented has better computation time than that of the known [19] in Relative Change (RC) with an average of 80.29% which is good for our implementation.

Chapter 6

CONCLUSION AND FUTURE WORK

This thesis is devoted to the study, design and implementation of the VS based on the use of proxy and oblivious signature for providing privacy to the voters of EVS [19]. This scheme is implemented with Windev Express 23 version for our voting application on Dell Inspiron 3542, Windows 8.1 Operating System and a RAM of 16 GB. EVS was studied, necessary for it proofs are provided, and also some adjustment in the oblivious signature of the EVS are made. Design, implementation and testing of the EVS are made in the thesis. Experiments with the EVS similar to those made in [19] are conducted.

Time of completion of six phases of the known EVS from [19] and our implementation are compared using Relative Change (RC). The implemented EVS computation time is better with about an average of 80.3% than the reported in [19].

For future work, since most of the computations were done on the voter's side, this might discourage the use of the system if implemented. A revised version of the protocol to have less computation on the voter's side will be encouraged. Furthermore, implementation of the modified protocol to use it in small-scale election will be given a consideration.

REFERENCES

[1] T.M. Buchsbaum, (2004) 'E-voting: international development and lessons learnt', Proceedings of the ESF TED Workshop on Electronic Voting in Europe, Schloss Hofen/Bregenz, Lake of Constance, Austria". [Online]. Available: https://www.researchgate.net/profile/Robert_Krimmer/publication/220789172_Security_Assets_in_E-Voting/links/0912f50cb2746f2e89000000.pdf#page=30. [Accessed 30 02 2019].

[2] P.J. Hayes, (1988). "Computer Architecture and Organization", (2nd edition) Textbook. Advances in Computers, vol.49, pp.286-289, 1999.

[3] R. Joaquin, P. Ferreira & C. Riberio , "EVIV: An End-to-End Verifiable Internet Voting System," Journal of Computer and Security, vol. 32, pp. 170-191, 2013.

[4] "Democracy". [Online]. Available: https://en.wikiquote.org/wiki/Democracy. [Accessed 20 02 2019].

[5] "Election 2016: Ballot mishaps across four states plague the AEC". [Online]. Avaialable:https://www.abc.net.au/news/2016-07-07/election-2016-widespread-ballot-issues-around-australia/7577548. [Accessed 03 03 2019].

[6] “Despairing about elections? This is why your vote matters”. [Online].Available: https://theconversation.com/despairing-about-elections-this-is-why-your-vote-matters-60123. [Accessed 03 03 2019].

[7] D. DeSilver, “U.S. voter turnout trails most developed countries”. Fact Tank Blog, Pew Research Center. August 2, 2016. [Online].Available: http://www.pewresearch.org/fact-tank/2018/05/21/u-s-voter-turnout-trails-most-developed-countries. [Accessed 03 03 2019].

[8] B.C. Burden, D. T. Canon & K. R. Mayer , et al, Election laws, mobilization, and turnout: The unanticipated consequences of election reform. American Journal of Political Science. September 9, 2013. AJAPS.12063.vol.58, Issue1, pp. 95-109, January 2014.

[9] "Why it takes so long to get election night results," Nov 6, 2018. [Online].Available: https://www.vox.com/2018/11/6/18066350/midterm-elections-2018-vote-counting. [Accessed 03 03 2019].

[10] V. Mateu, F. Sebe & M. Valls, "Constructing Credential-Based E-Voting Systems from Offline E-Coin Protocols," Journal of Network and Computer Applications, vol. 42, pp. 39-44, 2014.

[11] “Issue voting”. [Online].Available: https://en.wikipedia.org/wiki/Issuevoting. [Accessed 03 03 2019].

[12] M. M. Sarker & T. N. Akhund.” The Roadmap to the Electronic Voting System Development: A Literature Review”, IJAEMS, vol. 2, Issue 5, pp.492-497,2016.

[13] “Vote Recorder”. [Online].Available: http://edison.rutgers.edu/vote.htm. [Accessed 03 03 2019].

[14] "Historical Timeline, Electronic Voting Machines and Related Voting Technology”. [Online].Available: https://votingmachines.procon.org/view.timeline.php?timelineID=000021. [Accessed 21 02 2019].

[15] "Which Countries Use Electronic Voting?" [Online]. Available: https://www.lifewire.com/which-countries-use-electronic-voting-4174877. [Accessed 21 02 2019].

.

[16] M. O’ Meara (2013), “ Survey & Analysis of E-Voting Solutions”, A thesis submitted at the University of Dublin, Trinity College, May, 2013. [Online]. Available: https://scss.tcd.ie/publications/theses/diss/2013/TCD-SCSS-DISSERTATION-2013-045.pdf. [Accessed 11 02 2019].

[17] D. Jefferson, A. D. Rubin & B. Simons, et al (2004) “A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)” January 5, 2004. . [Online]. Available: http://euro.ecom.cmu.edu/program/courses/tcr17-803/MinorityPaper.pdf. [Accessed 11 02 2019].

[18] L.F. Cranor, (2003) In Search of the Perfect Voting Technology: No Easy Answers. In: Gritzalis D.A. (eds) Secure Electronic Voting. Advances in Information Security, vol. 7. pp.17-30, 2003.

[19] S.-Y. Chiou, T.-J, Wang & J,-M, Chen, "Design and Implementation of a Mobile Voting System Using a Novel Oblivious and Proxy Signature", Security and Communication Networks, vol. 2017, article Id 30752210, pp.1-16, 2017.

[20] M. M. Sarker, T. N. Akhund,"The Roadmap to the Electronic Voting System Development: A Literature Review". International Journal of Advanced Engineering, Management and Science (ISSN: 2454-1311), vol.2, issue5, pp. 492-497, 2016.

[21] A. Parveen, S. Habib & S. Sarwar "Scope and Limitation of Electronic Voting System" IJCSMC, vol. 2, issue 5, pp. 123-128, 2013.

[22] D. W. Jones (2003): "A Brief Illustrated History of Voting". [Online].Available: www.cs.uiowa.edu/~jones/voting. [Accessed 11 02 2019].

[23] H. Thompson (2006): "Expert Calls for increased E-Voting Security,". [Online].Available:https://www.computerworld.com/article/2560901/security0/expert-calls-for-increased-e-voting-security.html. [Accessed 11 02 2019].

[24] L. Norden, "Voting System Failures: A Database Solution", Brennan Center for Justice, Sep. 13, 2010. [Online].Available: https://www.brennancenter.org/publication/voting-system-failures-database-solution. [Accessed 20 02 2019].

[25] G. Miller & A. Entous,"Declassified report says Putin 'ordered' effort to undermine faith in U.S. election and help Trump". [Online].Available: https://www.washingtonpost.com/world/national-security/intelligence-chiefs-expected-in-new-york-to-brief-trump-on-russian-hacking/2017/01/06/5f591416-d41a-11e6-9cb0-54ab630851e8_story.html?noredirect=on&utm_term=.cac4a3087770. [Accessed 21 02 2019].

[26] F. Fleitz, "Was Friday's declassified report claiming Russian hacking of the 2016 election rigged?". [Online]. Available: https://www.foxnews.com/opinion/was-fridays-declassified-report-claiming-russian-hacking-of-the-2016-election-rigged. [Accessed 21 02 2019].

[27] E. Kurt ,"Trump, Putin and the hidden history of how Russia interfered in the U.S. presidential election" January 10, 2017). [Online].Available: https://www.newsweek.com/trump-putin-russia-interfered-presidential-election-541302. [Accessed 21 02 2019].

[28] W. Diffie & M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, pp. 644-654, 1976

[29] S. Davtyan, A. Kiayias & L. Michel, et al," Integrity of Electronic Voting Systems: Fallacious use of Cryptography," in Symposium On Applied Computing, SAC'12, Riva del Garda (Trento), Italy, March 25-29, 2012. [Online].Available:http://www.engr.uconn.edu/~sad06005/pubs/Conference/sac12.pdf. [Accessed 21 02 2019]

[30] "Cryptography". [Online].Available: https://en.wikipedia.org/wiki/Cryptography. [Accessed 21 02 2019]

[31] A. Dollin, "A comparison of a public and a secret key cryptosystem", (1995 February, 29). [Online]. Available: www.dcs.ed.ac.uk/home/adamd/essays/crypto.html. [Accessed 21 02 2019]

[32] "Symmetric Encryption".[Online]. Available: https://www.ssl2buy.com/wiki/wp-content/uploads/2015/12/Symmetric-Encryption.png. [Accessed 02 01 2019].

[33] "Asymmetric Encryption".[Online]. Available: https://www.ssl2buy.com/wiki/wp-content/uploads/2015/12/Asymmetric-Encryption.png. [Accessed 02 01 2019].

[34] J. S. Chou, "A novel k-out-of-n oblivious transfer protocol from bilinear pairing," Advances in Multimedia, vol. 2012, article ID 630610, pp. 1-9, Article ID 630610, 2012.

[35] L. Chen, “Oblivious signatures,” in Computer Security– ESORICS 94, Lecture Notes in Computer Science 875, vol. 875 of Lecture Notes in Computer Science, pp. 161–172, 1994.

[36] S.Y. Chiou, T.J, Wang and J.M, Chen ” Design and Implementation of a Multiple-Choice E-voting Scheme on Mobile System using Novel t -out-of- n Oblivious Signature”, Journal. Inf. Sci. Eng., 2018, vol. 34, pp. 135-154, 2018

[37] “Creating the setup program and deploying it: How to proceed?”. [Online]. Available: https://doc.windev.com/?2028001. [Accessed 03 03 2019]

[38] “WINDEV; Easily Develop Cross-platform Applications”. [Online]. Available: https://www.easi.net/business-software/solutions/windev-development-cross-platform. [Accessed 03 03 2019].

[39] “Deploying the Application”. [Online]. Available: https://doc.windev.com/en-US/?1410086701&name=_lesson_414_deploying the application. [Accessed 02 03 2019].

[40] “WINDEV Tools”. [Online]. Available: https://help.windev.com/?3084011&name=windev_tools. [Accessed 27 02 2019].

[41] "First U.S. Presidential Election". [Online]. Available: https://www.history.com/this-day-in-history/first-u-s-presidential-election. [Accessed 10 02 2019].

[42] "Relative Change and Difference". [Online]. Available: https://en.wikipedia.org/wiki/Relative_change_and_difference. [Accessed 24 06 2019]

APPENDICES

Appendix A. Voting System Source Codes

//The codes use French language terms. So we give their translation in Table A.1

Table A.1. English translations of French terms used in the source codes

French term	**English translation**
Si	**If**
Fin	**End**
TANTQUE	AS LONG AS
Alors	Then
Vrai	True
Info	Message
Reprisesaisie	Return To Capture;
RETOUR	RETURN
Faux	False
Raz	Reset
Grise	Grey
Titre	Title
Ligne	Line
Ouvre	Open
et	and
non	no

Appendix A.1. System Setup Phase Source Code

Appendix A.1.1 Publish BB

```
Click Publish// BB is published  “Publish- //p,q,g,h,yA” is clicked
Libellé1=Libellé1+" p = "+enter_p+", q = "+enter_q+", g = "+enter_g+" , h = "+enter_h+" , yA = "+Ya
Libellé1..Visible=Vrai
//end of click publish
```

Appendix A.1.2 Prime Number p

```
Exit from enter_p//Procedure invoked when cursor exits enter_p field
I est un entire
I=2;
TANTQUE (i<=(enter_p)-1)
        SI ((enter_p)modulo(i))<>0 ALORS
                i=i+1
                SI NON
                        Info("Please enter a prime number")
                        RepriseSaisie(enter_p)
                        RETOUR
            FIN
FIN
// code to perform p-1 divide q and to check validity
SI (enter_q<>0) ALORS
SI ((enter_p1)modulo(enter_q))<>0 ALORS
        Info("(P-1)/Q is a decimal number! please change their values ")
        RAZ(Vrai)
        RepriseSaisie(enter_p)
        RETOUR
ELSE
        enter_P_1_Q=enter_p1/enter_q
FIN
FIN
//end of exit from enter_p
```

Appendix A.1.3 Prime Number q

```
Exit from enter_q//Procedure invoked when cursor exits enter_q field
i1 est un entier
i1=2;
TANTQUE (i1<=(enter_q)-1)
        SI ((enter_q)modulo(i1))<>0 ALORS
```

```
                    i1=i1+1
            SINON
                    Info("Please enter a prime number")
                    RepriseSaisie(enter_q)
                    RETOUR
            FIN
FIN
TableSupprimeTout(Table1)
IF enter_p=0 THEN
            Info("Please enter the value of P")
            RepriseSaisie(enter_p)
END
Table1.Ligne2..Titre=enter_h
SI (enter_p<>0) ALORS
            SI ((enter_p1)modulo(enter_q))<>0 ALORS
                    Info("(P-1)/Q is a decimal number! please change their values ")
                    RAZ(Vrai)
                    RepriseSaisie(enter_p)
                    RETOUR
            ELSE
                    enter_P_1_Q=enter_p1/enter_q
            FIN
FIN
//end of exit from enter_q
```

Appendix A.1.4 Generator h Order p

```
whenever modifying enter_h//procedure to get h
i is int
FOR i=1 TO enter_p1 STEP 1

            TableAddLine(Table1,i,(Power(enter_h,i))modulo(enter_p))
            TableDisplay(Table1)
END
Table1..Visible=Vrai
// end procedure for h
```

Appendix A.1.5 Generator g Order p

```
whenever modifying enter_g//procedure to get g
Table2.Ligne2..Titre=enter_g
TableSupprimeTout(Table2)
IF enter_p=0 THEN
            Info("Please enter the value of P")
            RepriseSaisie(enter_p)
END
i is int
FOR i=1 TO enter_p1 STEP 1
            TableAddLine(Table2,i,(Power(enter_g,i))modulo(enter_p))
            TableDisplay(Table2)
END
Table2..Visible=Vrai
//exit enter_g
```

Appendix A.1.6 Compute y_A

```
whenever modifying Xa// procedure to get Ya
Ya=(Power(enter_g,Xa))modulo(enter_p)
//exit Xa
```

Appendix A.1.7 Prime Number p_B

```
Exit from enter_p_1 // procedure for entering Pb
i est un entier
i=2;
TANTQUE (i<=(enter_p_1)-1)
            SI ((enter_p_1)modulo(i))<>0 ALORS
                    i=i+1
                    SINON
                                Info("Please enter a prime number")
                                RepriseSaisie(enter_p_1)
                                RETOUR
            FIN
FIN
SI enter_q_1<>"" ALORS
```

```
        NB=enter_p_1*enter_q_1;
FIN
SI (enter_p_1<>0) ET (enter_q_1<>0) ALORS
        enter_n=(enter_p_1-1)*(enter_q_1-1);
FIN
//exit from Pb
```

Appendix A.1.8 Prime Number q_B

```
Exit from enter_q_1  //procedure to invoke when cursor exits enter_q_1
i1 est un entier
i1=2;
TANTQUE (i1<=(enter_q_1)-1)
        SI ((enter_q_1)modulo(i1))<>0 ALORS
                i1=i1+1
        SINON
                Info("Please enter a prime number")
                RepriseSaisie(enter_q_1)
                RETOUR
        FIN
FIN

SI enter_p_1<>"" ALORS
        NB=enter_p_1*enter_q_1;
FIN
SI (enter_p_1<>"") ET (enter_q_1<>"") ALORS
        enter_n=(enter_p_1-1)*(enter_q_1-1);
FIN
//TableSupprimeTout(Table1)
IF enter_p_1=0 THEN
        Info("Please enter the value of PB")
        RepriseSaisie(enter_p_1)
END
// end of exit enter Qb
```

Appendix A.1.9 Compute y_B

```
whenever modifying Xb// procedure to get Yb
Yb=(Power(enter_g,Xb))modulo(enter_p)
//end of exit for Xb
```

Appendix A.1.10 Compute d_B

```
Exit from Eb// procedure for checking Eb
r,r1,u,v,u1,v1,rs,us,vs sont des entiers;
r= Eb; r1= enter_n; u= 1; v= 0; u1= 0; v1= 1;
q est un entier
r = (Eb*u)+(enter_n*v); r1=( Eb*u1)+(enter_n*v1);
TANTQUE (r1 <> 0)
        q= r/r1;
        rs= r; us= u; vs= v;
        r= r1; u= u1; v= v1;
        r1= rs - q*r1; u1= us - q*u1; v1= vs - q*v1;
        FIN
SI (r<>1) ALORS
        Info("GCD is different from 1, please change the value")
        RepriseSaisie(Eb)
        RETURN
FIN
IF (u<0) ALORS
u=enter_n+u;
FIN
Db=u;
// end of exit from Eb
```

Appendix A.1.11 Publish N_B, y_B, e_B

```
Click Bouton1// procedure to publish the values of Nb, Eb, and Yb
Libellé3=Libellé3+"  NB = "+NB+", eB = "+Eb+" , yB = "+Yb
Libellé3..Visible=Vrai
//end of Click Bouton1
```

Appendix A.1.12 Enter x_A

```
SI (Xa<1) OR (Xa>(EXPRESS_crypt.enter_q-1)) ALORS
        Info("Please enter value between 1 and  "+(EXPRESS_crypt.enter_q-1))
```

```
        RepriseSaisie(Xa)
FIN
// end of exit from Xa
```

Appendix A.2. Proxy Phase Source Code

Appendix A.2.1 Publish BB

```
Initializing Libelle2 // to display published in bulletin board
Libellé2=("[eB ,NB ] ="+"["+EXPRESS_crypt.Eb+", "+EXPRESS_crypt.NB+" ] ")
//End of initializing Libelle2
```

Appendix A.2.2 Compute s_A, y'_p,

```
Exit from sai_k// on cursor exit
SI (sai_k<1) OR (sai_k>(EXPRESS_crypt.enter_/q-1)) ALORS
        Info("Please enter value between 1 and  "+(EXPRESS_crypt.enter_q-1))
        RepriseSaisie(sai_k)
FIN
Ra=Puissance((EXPRESS_crypt.enter_g),sai_k)modulo(EXPRESS_crypt.enter_p)
Sa=((EXPRESS_crypt.Xa)*(Ra)+(sai_k))modulo(EXPRESS_crypt.enter_q)
Libellé1=("[rA ,sA ] ="+"["+Ra+", "+Sa+" ] ")
Libellé1..Visible=Vrai
SI sai_k<>"" ALORS
        dlg=(Puissance(Ra,EXPRESS_crypt.Eb)modulo(EXPRESS_crypt.NB))
FIN
SI sai_k<>"" ALORS
        dlg1=(Puissance(Sa,EXPRESS_crypt.Eb)modulo(EXPRESS_crypt.NB))
FIN
yP=(Puissance((EXPRESS_crypt.enter_g),Sa)modulo(EXPRESS_crypt.enter_p))
//end of  Exit from sai_k
```

Appendix A.2.3 Compute $s_{A,B}$

```
Click SEND // clicked to produce result
monchono1, monchrono2,monchrono sont des entiers
desdlg=dlg
desdlg1=dlg1
gSA=yP
ChronoDébut()
gSA1=(Ra*(Puissance(EXPRESS_crypt.Ya,Ra)))modulo(EXPRESS_crypt.enter_p)
SI  (gSA=gSA1) ALORS
        SI YesNo("(g^sA)MOD(P)=(rA*yA^rA)MOD(P)= "+gSA+ "HOLDS"+"  Would you like to continue
the process")=Vrai ALORS
                                        Sp=((Sa)+(EXPRESS_crypt.Xb))modulo(EXPRESS_crypt.enter_q)
                monchono1=ChronoFin()
        ELSE
                RepriseSaisie(sai_k)
                        FIN
ELSE
        Info("Values are not equal")
FIN
ChronoDébut()
a1,b1,mod,div,a2 sont des entiers
mod=(EXPRESS_crypt.Db)modulo(5)
div=PartieEntière(EXPRESS_crypt.Db/5)
a2=(Puissance(EXPRESS_crypt.enter_g,Sa))modulo(EXPRESS_crypt.enter_p)
a1=(Puissance(a2,div))modulo(EXPRESS_crypt.NB)
b1=(Puissance(a2,mod))modulo(EXPRESS_crypt.NB)
S_A_B=((Puissance(a1,5))*b1)modulo(EXPRESS_crypt.NB)
monchrono2=ChronoFin()
Libellé3="[ yp' = "+yP+" ]"
monchrono=(monchono1+monchrono2)/1000
Saisie1=monchrono
Saisie3=Saisie1+ Saisie2
//End of Click SEND
```

Appendix A.2.4 Comparism of S and H value

```
Click Bouton3//on click procedure
monchrono est un entier
ChronoDébut()
```

```
sab1,hyp sont des entiers
sab1=(Puissance(S_A_B,EXPRESS_crypt.Eb))modulo(EXPRESS_crypt.NB)
hyp=(yP)modulo(EXPRESS_crypt.NB)
S_A_B_eB=sab1
H_yP=hyp
H_yP=hyp
SI(sab1=hyp) ALORS
        Info("S(A,B)^eB=H(yP') = "+sab1+"  hold")
FIN
S_A_B_eB..Visible=Vrai
H_yP..Visible=Vrai
monchrono=ChronoFin()
Saisie2=(monchrono)/1000
Saisie3=Saisie3+Saisie2
//End of Bouton3
```

Appendix A.2.5 RSA Decryption for secret proxy signature

```
Click Bouton2// click to decrypt..
monchono1 sont des entiers
ChronoDébut()
desdlg2=POW_FUNC1(desdlg,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
desdlg3=POW_FUNC1(desdlg1,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
monchono1=ChronoFin()
Saisie5=(monchono1)/1000
Saisie4=Saisie5
//End of click Bouton2
```

Appendix A.2.6 Forward value $s_{A,B}$

```
Click Bouton1 // Send back to A
monchono2 sont des entiers
ChronoDébut()
S_A_B1=S_A_B
S_A_B1..Visible=Vrai
Bouton3..Visible=Vrai;
monchono2=ChronoFin()
Saisie6=(monchono2/1000)
Saisie4=Saisie4+Saisie6
//Endof Click buoton1
```

Appendix A.3. Registration Phase Source Code

Appendix A.3.1 Hash function for password

```
 Exit from ID //
SI ID<>"" ALORS
som est un entier
som=0;
SI (Password<>"") ALORS
        som=H_FUNC(Password)
        pwencrypt=som;
FIN
SI (Pseudoname="") AND (Password="") ALORS
        Libellé1="[ "+ID+","+0+", "+0+" ]"
FIN
SI (Pseudoname<>"") AND (Password="") ALORS
        Libellé1="[ "+ID+","+Pseudoname+", "+0+" ]"
FIN
SI (Pseudoname="") AND (Password<>"") ALORS
        Libellé1="[ "+ID+","+0+", "+som+" ]"
FIN
SI (Pseudoname<>"") AND (Password<>"") ALORS
        Libellé1="[ "+ID+","+Pseudoname+", "+som+" ]"
FIN
SI ID<>"" ALORS
        SI HLitRecherchePremier(EXPRESS_flag,ID,ID,hIdentique)=Vrai ALORS
                ID=EXPRESS_flag.ID
        FIN
FIN
```

```
FIN
```
//End of Exit from ID

Appendix A.3.2 Hash function for Pseudoname

```
Exit from pseudoname//this works by keypress or cursor blink
som est un entier
som=0;
SI (Password<>"")ALORS
som=H_FUNC(Password)
pwencrypt=som
FIN
SI (ID="") AND (Password="") ALORS
        Libellé1="[ "+0+","+Pseudoname+", "+0+" ]"
FIN
SI (ID<>"") AND (Password="") ALORS
        Libellé1="[ "+ID+","+Pseudoname+", "+0+" ]"
FIN
SI (ID="") AND (Password<>"") ALORS
        Libellé1="[ "+0+","+Pseudoname+", "+som+" ]"
FIN
SI (ID<>"") AND (Password<>"") ALORS
        Libellé1="[ "+ID+","+Pseudoname+", "+som+" ]"
FIN
SI ID<>"" ALORS
        SI HLitRecherchePremier(EXPRESS_flag,ID,ID,hIdentique)=Vrai ALORS
                PN=EXPRESS_flag.pseudonime
        FIN
FIN
//End of Exit from pseudoname
```

Appendix A.3.3 Encryption of Id, pn, pw

```
Click Bouton3//
monchrono est un entier
i est un entier
ch est une chaîne
ChronoDébut()
sai_ch=ENCRYP_FUNC(Pseudoname,EXPRESS_crypt.Eb,EXPRESS_crypt.NB)
POUR i=1 À Length(sai_ch) PAS 1
        SI (sai_ch[[i]]<>"-") ALORS
                ch=ch+sai_ch[[i]]
        FIN
FIN
pseu=ch
SI Libellé4<>"Encrypted Values :" ALORS
        Libellé4="Encrypted Values :"
        Libellé4=Libellé4+"[ "+POW_FUNC(ID,EXPRESS_crypt.Eb,EXPRESS_crypt.NB)+", "+pseu+",
"+POW_FUNC(pwencrypt,EXPRESS_crypt.Eb,EXPRESS_crypt.NB)+" ]"
monchrono=ChronoFin()
ELSE
Libellé4=Libellé4+"[ "+POW_FUNC(ID,EXPRESS_crypt.Eb,EXPRESS_crypt.NB)+", "+pseu+",
"+POW_FUNC(pwencrypt,EXPRESS_crypt.Eb,EXPRESS_crypt.NB)+" ]"
FIN
Saisie2=(monchrono)/1000
Saisie3=Saisie3+Saisie2
//End of Click bouton3
```

Appendix A.3.4 Decryption of pn, pw, id

```
Click Bouton2//
monchrono1 est un entier
id1,ps1 sont des entiers
pseudo est une chaîne
ChronoDébut()
id1=POW_FUNC1(Table1.ID,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
ps1=POW_FUNC1(Table1.PW,EXPRESS_crypt.Db,EXPRESS_crypt.NB)+(PartieEntière(pwencrypt/EXPRESS
_crypt.NB))*EXPRESS_crypt.NB
pseudo=DECRYPT_FUNC(sai_ch,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
 monchrono1=ChronoFin()
SI Libellé5<>"" ALORS
        Libellé5=""
        Libellé5=Libellé5+"[ "+id1+", "+pseudo+", "+ps1+" ]"
```

```
            Libellé5..Visible=Vrai
            ELSE
            Libellé5=Libellé5+"[ "+id1+", "+pseudo+", "+ps1+" ]"
            Libellé5..Visible=Vrai
            FIN
Saisie5=(monchrono1)/1000
Saisie4=Saisie4+Saisie5
//End of Click Buoton2
```

Appendix A.3.5 Decryption of pn

```
Click Bouton1//
monchrono2 est un entier
id1,ps1 sont des entiers
pseudo est une chaîne
ChronoDébut()
id1=POW_FUNC(Table1.ID,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
ps1=POW_FUNC(Table1.PW,EXPRESS_crypt.Db,EXPRESS_crypt.NB)+(PartieEntière(pwencrypt/EXPRESS_
crypt.NB))*EXPRESS_crypt.NB
pseudo=DECRYPT_FUNC(sai_ch,EXPRESS_crypt.Db,EXPRESS_crypt.NB)
monchrono2=ChronoFin()
Saisie6=(monchrono2)/1000
Saisie4=Saisie4+Saisie6
//End of Click Bouton1
```

Apprendix A.3.6 Store pn, id, pw in Dbase

```
Click Buoton4//
monchrono est un entier
ch sont des chaînes
ChronoDébut()
ch=Pseudoname+EXPRESS_crypt.Eb
SI (POW_FUNC1(sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))= (H_FUNC(ch))modulo(EXPRESS_crypt.NB)
ALORS
            SI test<>"" ALORS
            test=""
            test="[ sB^eB = H(pn,eB)Mod(NB) = "+ (H_FUNC(ch))modulo(EXPRESS_crypt.NB)
            ELSE
                        test="[ sB^eB = H(pn,eB)Mod(NB) = "+ (H_FUNC(ch))modulo(EXPRESS_crypt.NB)
            FIN
            Ouvre(EXPRESS_vote)
monchrono=ChronoFin()
ELSE
            SI test<>"" ALORS
                        test=""
                        test="[ sB^eB ="+(POW_FUNC1(sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))+"<>"+
"H(pn,eB)Mod(NB) = "+(H_FUNC(ch))modulo(EXPRESS_crypt.NB)+" ]"
            ELSE
                        test="[ sB^eB ="+(POW_FUNC1(sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))+"<>"+
"H(pn,eB)Mod(NB) = "+(H_FUNC(ch))modulo(EXPRESS_crypt.NB)+" ]"
            FIN
            Info("You can't vote")
FIN
Saisie1=(monchrono)/1000
Saisie3=Saisie3+Saisie1
//End of Click Buoton4
```

Appendix A.4. Circling Phase Source Code

Appendix A.4.1 Generate r

```
Initializing enter_r// initializing random value for r
InitHasard()     // it generate a random value for r
enter_r=Hasard(1,EXPRESS_crypt.enter_q-1)
// end of Initializing enter_r
```

Appendix A.4.2 Ecryption of R ballot

```
Click Bouton2 //
monchrono1 est un entier
ch est une chaîne
ChronoDébut()
ch=EXPRESS_registration.pwencrypt
ch=ch+enter_r1..ValeurAffichée
hr=H_FUNC(ch)
```

```
hr1=hr
monchrono1=ChronoFin()
Saisie8=(monchrono1)/1000
Saisie7=Saisie7+Saisie8
//end of click Buoton2
```

Appendix A.4.3 Choose blinding factor v

```
Initializing select_v//
InitHasard()
select_v=Hasard(1,EXPRESS_crypt.enter_q-1)
//end of Initializing selct_v
```

Appendix A.4.4 Completeness of Oblivious signature generated

```
Click Bouton7 //
monchrono2 est un entier
sig,ei,i,rs,yp,res sont des entiers
ch est une chaîne
ChronoDébut()
yp=POW_FUNC1(EXPRESS_proxy.yP*EXPRESS_crypt.Yb,1,EXPRESS_crypt.enter_p)
POUR i=1 À TableOccurrence(k)
sig=POW_FUNC1(EXPRESS_crypt.enter_g*EXPRESS_crypt.enter_h,i,EXPRESS_crypt.enter_p)
sig=(c*sig)modulo(EXPRESS_crypt.enter_p)
  ch=""
  rs=POW_FUNC1(EXPRESS_crypt.enter_g,k[i].si1,EXPRESS_crypt.enter_p)
  res=(rs*POW_FUNC1(yp,k[i].ei,EXPRESS_crypt.enter_p))modulo(EXPRESS_crypt.enter_p)
ch=Table1[i].Message+EXPRESS_registration.Pseudoname+POW_FUNC1(res*k[i].δi,1,EXPRESS_crypt.enter_
p)
  ei=H_FUNC(ch)
     TableAjouteLigne(k1,i,sig,ei)
  TableAffiche(k1)
FIN
monchrono2=ChronoFin()
Saisie9=(monchrono2)/1000
Saisie7=Saisie7+Saisie9
// end of click Bouton7
```

Appendix A.4.5 Flag (pn) value checker

```
Click Bouton4 //
monchrono2 est un entier
ChronoDébut()
SI (pn_hr_c)<>"" ALORS
            pn_hr_c=""
            pn_hr_c="[ "+EXPRESS_registration.Pseudoname+", "+hr+", "+c+" ]"
            pn_hr_c..Visible=Vrai
ELSE
            pn_hr_c="[ "+EXPRESS_registration.Pseudoname+", "+hr+", "+c+" ]"
            pn_hr_c..Visible=Vrai
FIN
SI hhr=hr1 ALORS
                        SI
HLitRecherche(EXPRESS_flag,EXPRESS_flag.ID,EXPRESS_registration.ID,hIdentique)=Vrai ALORS
                                                SI Libellé3="Flag(PN)=" ALORS
                                                Libellé3=Libellé3+EXPRESS_flag.flag
                                                Libellé3..Visible=Vrai
monchrono2=ChronoFin()
                                    ELSE
                                                Libellé3="Flag(PN)="
                                                Libellé3=Libellé3+EXPRESS_flag.flag
                                                Libellé3..Visible=Vrai
                                    FIN
FIN
FIN
Saisie8=(monchrono1)/1000
Saisie7=Saisie7+Saisie8
// end of bouton4
```

Appendix A.4.6 Compleness of the proxy signature check

```
click Buoton1//on click procedure
monchrono est un entier
enter_r1=enter_r
ch est une chaîne
```

```
ChronoDébut()
ch=EXPRESS_registration.pwencrypt
ch=ch+enter_r
hhr=H_FUNC(ch)
monchrono=ChronoFin()
Saisie5=(monchrono)/1000
Saisie4=Saisie4+Saisie5
//End of click Bouton1
```

Appendix A.4.7 Calculation of Ki for completion of signature

```
Click bouton5 //
monchrono1 est un entire
ChronoDébut()
SI TableOccurrence(k)+1<=Saisie2 ALORS
        i,ki1,sig,e1,s1,sig1,j sont des entiers
        ch est une chaîne
        i=TableOccurrence(k); j=1;
        SI TableOccurrence(k)<=Saisie2 ALORS
                ki1=POW_FUNC1(EXPRESS_crypt.enter_g,enter_ki,EXPRESS_crypt.enter_p)
        sig1=POW_FUNC1(EXPRESS_crypt.enter_g*EXPRESS_crypt.enter_h,i+1,EXPRESS_crypt.enter_p)
                sig=(c*sig1)modulo(EXPRESS_crypt.enter_p)
                                ch=""
ch=Table1[i+1].Message+EXPRESS_registration.Pseudoname+POW_FUNC1(ki1*sig,1,EXPRESS_crypt.enter_
p)
                                        e1=H_FUNC(ch)
                        s1=(enter_ki-EXPRESS_proxy.Sp*e1)modulo(EXPRESS_crypt.enter_q)
                IF s1<0 ALORS
                s1=s1+EXPRESS_crypt.enter_q
                FIN
                                TableAjouteLigne(k,i+1,ki1,sig,e1,s1)
                        ELSE
                Info("You have already entered the required number for ki")
                RETURN
                FIN
        TableAffiche(k)
monchrono1=ChronoFin()
ELSE
        Info("Your calculations are complete")
        RETURN
FIN
Saisie6=(monchrono1)/1000
Saisie4=Saisie4+Saisie6
// End of click bouton5
```

Appendix A.4.8 Flag (pn) validator

```
Click Bouton6 //
monchrono est un entire
ChronoDébut()
SI HLitRecherchePremier(EXPRESS_flag,EXPRESS_flag.ID,EXPRESS_registration.ID,hIdentique)=Vrai
ALORS
        EXPRESS_flag.flag=1
        HModifie(EXPRESS_flag)
        SI Libellé3="Flag(PN)=" ALORS
                Libellé3=Libellé3+EXPRESS_flag.flag
                Libellé3..Visible=Vrai
        ELSE
                Libellé3="Flag(PN)="
                Libellé3=Libellé3+EXPRESS_flag.flag
                Libellé3..Visible=Vrai
        FIN
FIN
i est un entier
POUR i=1 À TableOccurrence(k)
        TableAjouteLigne(Table2,"[ "+k[i].ei+", "+k[i].si1+" ]")
FIN
TableAffiche(Table2)
Table2..Visible=Vrai
monchrono=ChronoFin()
Saisie5=(monchrono)/1000
```

```
Saisie4=Saisie4+Saisie5
// End of click Bouton6
```
Appendix A.4.9 Final signature confirmation
```
Click Bouton8 //
monchrono2 est un entire
ChronoDébut()
comp_s=POW_FUNC1((k[choose_b].si1+EXPRESS_vote.select_v+(choose_b)),1,EXPRESS_crypt.enter_q)
comp_e=k[choose_b].ei
SI e_s<>"(mb)" ALORS
        e_s="(mb)"
        e_s="[ "+comp_e+","+comp_s+" ]"
        e_s..Visible=Vrai
        Libellé1..Visible=Vrai
monchrono2=ChronoFin()
ELSE
        e_s="[ "+comp_e+","+comp_s+" ]"
        e_s..Visible=Vrai
        Libellé1..Visible=Vrai
FIN
Saisie8=(monchrono2)/1000
Saisie7=Saisie7+Saisie8
```
// end of click Bouton8

Appendix A.5. Voting Phase Source Code

Appendix A.5.1 Signature confirmation
```
Click Bouton1//
monchrono2 est un entire
ChronoDébut()
hh_eb=EXPRESS_registration.keyr
mb=EXPRESS_vote.Table1[EXPRESS_vote.choose_b].Message+","+EXPRESS_vote.comp_e+","+EXPRESS_
vote.comp_s
monchrono2=ChronoFin()
Saisie1=(monchrono2)/1000
Saisie3=Saisie3+Saisie1
//end of Click Bouton1
```
Appendix A.5.2 Symmetric Encryption to produce (Cert(R),Cr)
```
Click Bouton3//
monchrono est un entier
i est un entier
ch est une chaîne
ChronoDébut()
get_cr=""
TANTQUE i<(Length(mb)+1)
        ch=""
        TANTQUE (mb[[i]]<>",") ET (i<(Length(mb)+1))
                ch=ch+mb[[i]]
                i=i+1;
        FIN
        i=i+1
        SI get_cr<>"" ALORS
        get_cr=get_cr+","+POW_FUNC1(Val(ch),hh_eb,EXPRESS_crypt.NB)
   ELSE
        get_cr=POW_FUNC1(Val(ch),hh_eb,EXPRESS_crypt.NB)
        FIN
FIN
cert_r="([ "+EXPRESS_registration.Pseudoname+", "+EXPRESS_crypt.Eb+", "+EXPRESS_registration.sb+"
])"+",(["+get_cr+"])"
monchrono=ChronoFin()
Saisie2=(monchrono)/1000
Saisie3=Saisie3+Saisie2
// end of Click Bouton3
```
Appendix A.5.3 Sending of (Cert(R),Cr)
```
Click Bouton2//
monchrono2 est un entire
ChronoDébut()
cert_r1=cert_r
```

```
monchrono2=ChronoFin()
Saisie1=(monchrono2)/1000
Saisie3=Saisie3+Saisie1
// end of Click Bouton2
```

Appendix A.5.4 Verification of (Cert(R),Cr)

```
Click Bouton4//
monchrono1 est un entier
ch sont des chaînes
ChronoDébut()
ch=EXPRESS_registration.Pseudoname+EXPRESS_crypt.Eb
IF (POW_FUNC1(EXPRESS_registration.sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))=
(H_FUNC(ch))modulo(EXPRESS_crypt.NB) ALORS
SI test<>"" ALORS
        test=""
        test="[ sB^eB = H(pn,eB)Mod(NB) = "+ (H_FUNC(ch))modulo(EXPRESS_crypt.NB)
ELSE
        test="[ sB^eB = H(pn,eB)Mod(NB) = "+ (H_FUNC(ch))modulo(EXPRESS_crypt.NB)
FIN
 Info("HOLD")
ELSE
SI test<>"" ALORS
        test=""
        test="[ sB^eB
="+(POW_FUNC1(EXPRESS_registration.sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))+"<>"+
"H(pn,eB)Mod(NB) = "+(H_FUNC(ch))modulo(EXPRESS_crypt.NB)+" ]"
ELSE
        test="[ sB^eB
="+(POW_FUNC1(EXPRESS_registration.sb,EXPRESS_crypt.Eb,EXPRESS_crypt.NB))+"<>"+
"H(pn,eB)Mod(NB) = "+(H_FUNC(ch))modulo(EXPRESS_crypt.NB)+" ]"
FIN
Info("IT DOESN'T HOLD")
FIN
monchrono1=ChronoFin()
Saisie5=(monchrono1)/1000
Saisie4=Saisie4+Saisie5
// end of Click Bouton4
```

Appendix A.5.5 Sending of (Cert(R),Cr) to BB

```
Click Bouton5//
monchrono2 est un entire
ChronoDébut()
SI test1<>"" ALORS
        test1=""
        test1=test1+cert_r1
ELSE
        test1=cert_r1
FIN
monchrono2=ChronoFin()
Saisie6=(monchrono2)/1000
Saisie4=Saisie4+Saisie6
//End of Click Bouton5
```

Appendix A.6. Counting Phase Source Code

Appendix A.6.1 Get key generated

```
Click Bouton1//
hh_eb1=hh_eb2
cr1=EXPRESS_Fenêtre1.get_cr
//end of Click Bouton1
```

Appendix A.6.2 Symmetric key decryption

```
Click Bouton2//
monchrono1 est un entier
ChronoDébut()
decrypt_cr=DECRYPT_FUNC1(cr1,Inverse_MOD_FUNC(hh_eb1,EXPRESS_crypt.enter_p),EXPRESS_crypt.e
nter_p)
decrypt_cr=EXPRESS_Fenêtre1.mb
monchrono1=ChronoFin()
Saisie1=(monchrono1)/1000
Saisie3=Saisie3+Saisie1
```

```
//end of click Bouton2
```
Appendix A.6.3 Publishing on BB
```
Click Bouton3//
monchrono1 est un entire
ChronoDébut()
test1="( "+EXPRESS_Fenêtre1.cert_r+",[ "+EXPRESS_vote.Table1[EXPRESS_vote.choose_b].Message+"
],["+hh_eb1+"])"
monchrono1=ChronoFin()
Saisie2=(monchrono1)/1000
Saisie3=Saisie3+Saisie2
//end of click Bouton3
```
Appendix A.6.4 Calculate Proxy public key y_p
```
Click Bouton4//
monchrono2 est un entier
ChronoDébut()
get_yp=POW_FUNC1(EXPRESS_proxy.yP*EXPRESS_crypt.Yb,1,EXPRESS_crypt.enter_p)
monchrono2=ChronoFin()
Saisie1=(monchrono1)/1000
Saisie3=Saisie3+Saisie1
//end of click Bouton4
```
Appendix A.6.5 Signature verification
```
Click Bouton5//
monchrono2 est un entier
comp_e=EXPRESS_vote.comp_e
ch est une chaîne
rs,res sont des entiers
ChronoDébut()
ch=""
rs=POW_FUNC1(EXPRESS_crypt.enter_g,EXPRESS_vote.comp_s,EXPRESS_crypt.enter_p)
res=(rs*POW_FUNC1(get_yp,EXPRESS_vote.comp_e,EXPRESS_crypt.enter_p))modulo(EXPRESS_crypt.ente
r_p)
ch=ch+EXPRESS_vote.Table1[EXPRESS_vote.choose_b].Message
ch=ch+EXPRESS_registration.Pseudoname
ch=ch+res
comp_e1=H_FUNC(ch)
IF comp_e=comp_e1 THEN
        Info("The signature is valid and the ballot is counted ")
        Bouton6..Visible=Vrai
ELSE
        Info("The signature is invalid ")
        RETURN
END
monchrono2=ChronoFin()
Saisie2=(monchrono2)/1000
Saisie3=Saisie3+Saisie2
//end of click Bouton5
```
Appendix A.6.6 Publish election result on BB
```
Click Bouton6//
monchrono2 est un entire
ChronoDébut()
SI test3<>"" ALORS
        test3=""
        test3="The ballot is counted, "
ELSE
        test3="The ballot is counted, "
FIN
monchrono2=ChronoFin()
Saisie1=(monchrono2)/1000
Saisie3=Saisie3+Saisie1
//end of click Bouton6
//-----------------------------------------------------------------------------------------
//Global Procedure H_FUNC
//-------------------
PROCEDURE H_FUNC(plaint est une chaîne)
som, i sont des entiers
som=0;
```

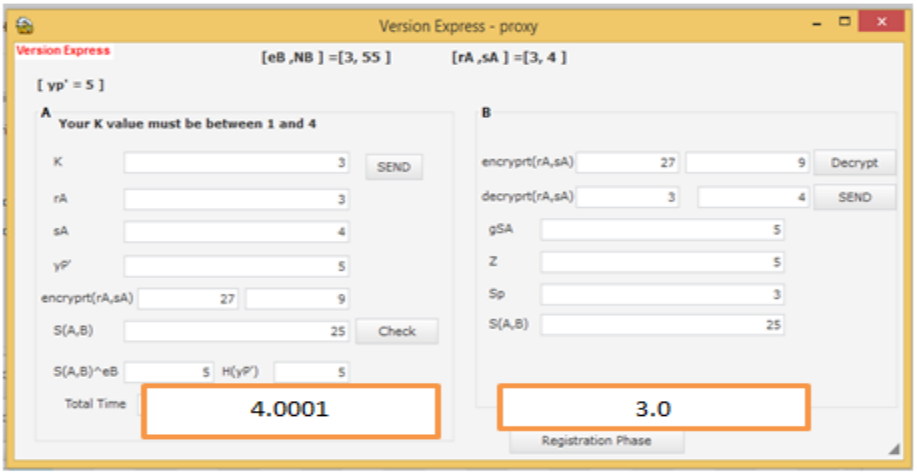
```
POUR i=1 À Length(plaint) PAS 1
     som=som+Asc(plaint[[i]])modulo(256)
     som=(som)modulo(256)
FIN
RENVOYER som
//-----------------------------------------------------------------------------------------------------------------
-
//PROCEDURES
Global Procedure ENCRYP_FUNC  // Function call for  Encryption
PROCEDURE ENCRYP_FUNC(ch est une chaîne,eb est un entier, nb est un entier )
i,res sont des entiers
Val est une chaîne
Val=""
 POUR i=1 À Length(ch) PAS 1
        SI
HLitRecherchePremier(EXPRESS_alphabet,EXPRESS_alphabet.letter,ch[[i]],hIdentique)=V
rai ALORS
               res=(POW_FUNC1(EXPRESS_alphabet.number,eb,nb))
               SI (Val="") ALORS
                  Val=Val+NumériqueVersChaîne(res)
               SINON
                  Val=Val+"-"+NumériqueVersChaîne(res)
                FIN
 FIN

 FIN
 RENVOYER Val
//-----------------------------------------------------------------------------------------------------------------
-
Global Procedure DECRYPT_FUNC // Fucntion for Decrytion
PROCEDURE DECRYPT_FUNC(ch est une chaîne ,db est un entier, nb est un entier )
i,nb1 sont des entiers
ch1,me sont des chaînes
i=1
TANTQUE i<(Length(ch)+1)
               ch1=""
       TANTQUE (ch[[i]]<>"-") ET (i<(Length(ch)+1))
               ch1=ch1+ch[[i]]
               i=i+1;
       FIN
                nb1=POW_FUNC1(Val(ch1),db,nb)
         SI
HLitRecherchePremier(EXPRESS_alphabet,EXPRESS_alphabet.number,nb1,hIdentique)=Vrai
ALORS
               me=me+EXPRESS_alphabet.letter
         FIN
       SI (ch[[i]]="-") OU (i<=Length(ch)) ALORS
               i=i+1
       FIN
FIN
RENVOYER me
//-----------------------------------------------------------------------------------------------
Global Procedure POW_FUNC1 //  Function for  raising  to power
PROCEDURE POW_FUNC1(nb est un entier,e est un entier, mod est un entier)
i, res sont des entiers
res=1;
POUR i=1 À e PAS 1
       res=(res*nb)modulo(mod)
FIN
RENVOYER res
//-----------------------------------------------------------------
Global Procedure DECRYPT_FUNC1
 PROCEDURE DECRYPT_FUNC1(ch est une chaîne ,db est un entier, nb est un entier )
i,nb1 sont des entiers
ch1,me sont des chaînes
```

```
i=1
TANTQUE i<(Length(ch)+1)

        ch1=""
        TANTQUE (ch[[i]]<>",") ET (i<(Length(ch)+1))
                ch1=ch1+ch[[i]]
                i=i+1;
        FIN
        nb1=POW_FUNC1(Val(ch1),db,nb)
        SI (me<>"") ALORS
                me=me+","+nb1
        ELSE
                me=nb1
        FIN

        SI (ch[[i]]=",") OU (i<=Length(ch)) ALORS
                i=i+1
        FIN
        FIN
RENVOYER me
```

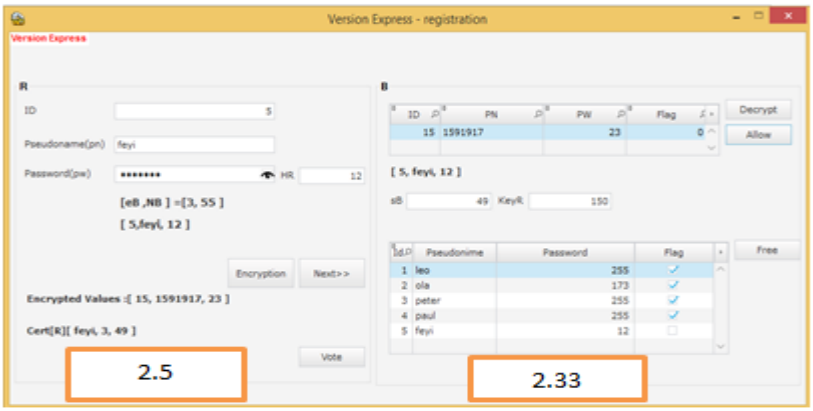
```
Global Procedure Inverse_MOD_FUNC
PROCEDURE Inverse_MOD_FUNC(nb est un entier,mod est un entier)
r,r1,u,v,u1,v1,rs,us,vs sont des entiers;
r= nb; r1= mod; u= 1; v= 0; u1= 0; v1= 1;
q est un entier
r = (nb*u)+(mod*v); r1=( nb*u1)+(mod*v1);
TANTQUE (r1 <> 0)
        q= r/r1;
        rs= r; us= u; vs= v;
        r= r1; u= u1; v= v1;
        r1= rs - q*r1; u1= us - q*u1; v1= vs - q*v1;
        FIN
IF (u<0) ALORS
        u=mod+u;
FIN
RENVOYER u;
```

Appendix B. Experimental Results

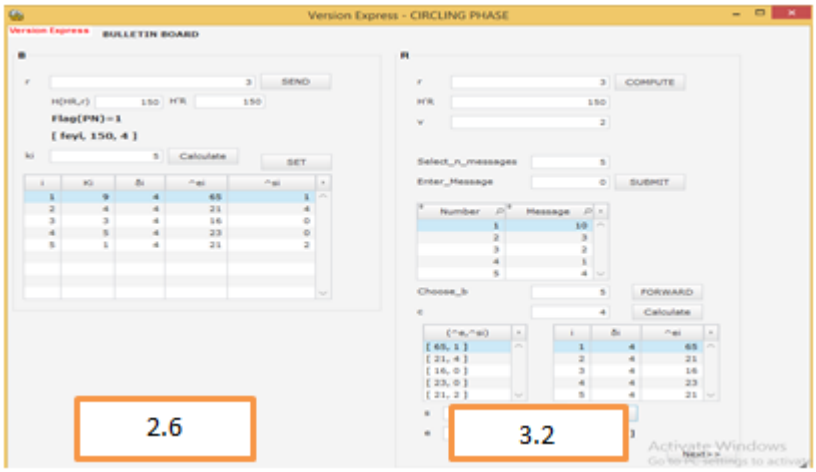
The Screen shot for Computation Time:



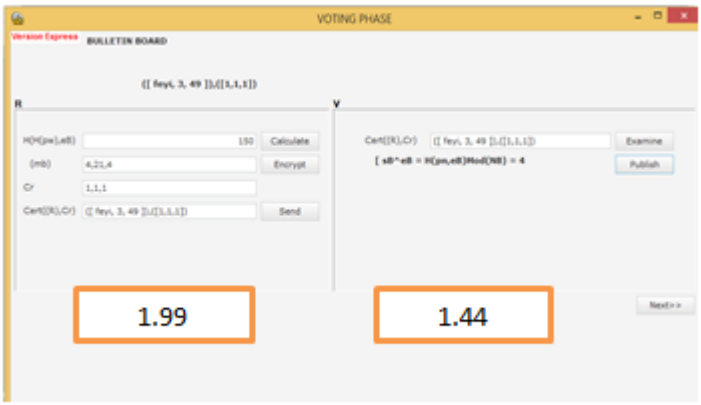
Screen shot for Computation Time for Proxy Phase (PP) of A and B, see (Appendix A, A.2 proxy phase source code, lines #177,181,194,202,204-206, 210,222-224,227-228,231-234,236-237, 241-243).



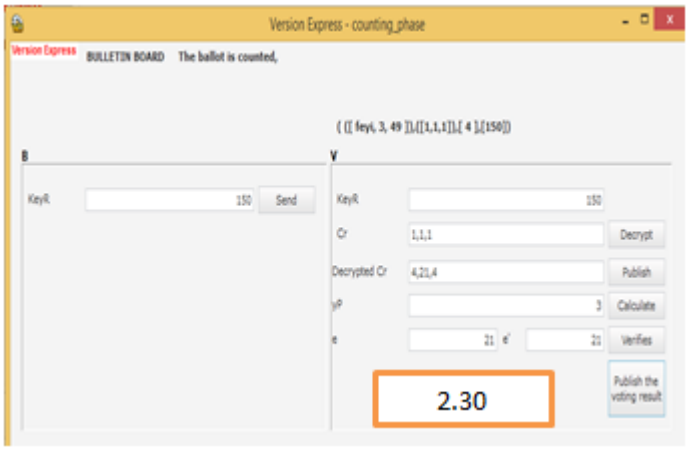
Screen shot for Computation Time for Register Phase (RP) of R and B, see (Appendix A, A.3 registration phase source code, lines# 298,301,313,318-319,322,325,330,339-340,343,346,351-353,356,358,369,381-382)



Screen shot for Computation Time for Circling Phase (CiP) of B and R, see (Appendix A, A.4 circling phase source code, lines# 389,392,396-398,407-408,430,435-437,439-440,459-461,464,466,471-473,480-481,486,491-492,495-496,511,519-520,523,526,539-541,544-545,553,559-560).



Screen shot for Computation Time for Voting Phase (VP) of R and V, see (Appendix A, A.5 voting phase source code, lines# 563-564,568-570, 573,576,593-595,598-599,601-603,606,608,632-634,637-638,645-647).



Screen shot for Computation Time for Counting Phase (CoP) of V,

see (Appendix A, A.6 counting phase source code, lines# 654-655,659-661,664-665,668-670,673-674,676-678,681,685,700-702,705-706,713-715). Verifier makes the output.