

A Secure Electronic Voting Scheme Based on Evotx-MA and REVS E Voting Blind Signature Protocols

By

Reem Ali Al-Saidi

Supervised By

Prof. Nidal Shilbayeh

Master Thesis

**Submitted in Partial Fulfillment of the
Requirements for the Master Degree
In Computer Science**

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman – Jordan

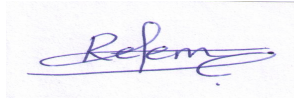
July, 2011

Middle East University
Authorization Statement

I, Reem Ali Nazmi Al-Saidi, authorize Middle East University to supply copies of my thesis to libraries, establishments or individuals upon their request, according to the university regulations.

Name: Reem Ali Al-Saidi.

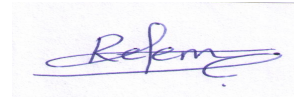
Signature:



جامعة الشرق الاوسط

اقرار تفويض

أنا ريم علي الصعيدي أفوض جامعة الشرق الاوسط بتزويد نسخ من رسالتي للمكتبات
او المؤسسات او الهيئات او الافراد عند طلبها .



التوقيع :

Middle East University

Examination Committee Decision

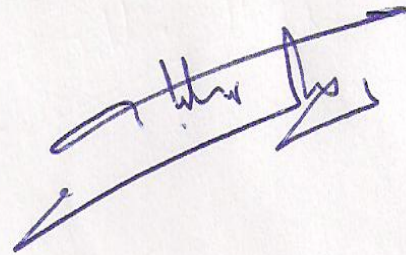
This is to certify that the thesis entitled "**A Secure Electronic Voting Scheme Based on Evox-MA and REVS E Voting Blind Signature Protocols**" was successfully defined and approved on 2011.

Examination Committee Members

Signature

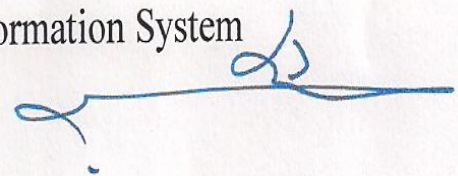
- Prof. Nidal Shilbayeh

Professor, Department of Computer Science
(Middle East University)



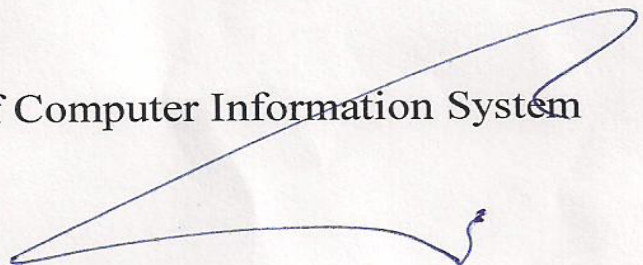
- Dr. Zeyad Al-Fwaeer

Assistant Professor, Department of Computer Information System
(Middle East University)



- Dr. Mohammad Alia

Assistant Professor, Department of Computer Information System
(Al-Zaytoonah University)



Dedication

To my lovely Parents

Acknowledgments

"Sometimes our light goes out but is blown into flame by another human being .Each of us sends his deepest thanks to those who have rekindled this light"....

I would like to thank, Prof.Nidal Shilbayeh who has supported me with his encouragements and many fruitful conversations which paved the way for the successful completion of this thesis.

Also, I would like to especially thank my family for supporting me all the time, without them nothing would have been possible.

Table of contents

Authorization Statement.....	II
Examination Committee Decision	III
Dedication	IV
Acknowledgement	V
List of Figures.....	IX
List of Tables	XI
Abbreviations.....	XII
Abstract.....	XII
الملخص	XIV

Chapter One: Introduction.....1

1.1 Introduction.....	1
1.2 Problem Statement.....	4
1.3 Research Objectives.....	6
1.4 Motivation.....	7
1.5 Significance of Research	8
1.6 Thesis Organization.....	9

Chapter Two: Literature Review.....10

2.1 What is E voting?.....	10
2.2 E voting Phases.....	12
2.3 E voting General Requirements.....	14
2.4 E voting Schemes.....	16
2.4.1 EVS based on Blind Signature.....	16
2.4.1.1 Implementation of blind signature Protocol in EVS.....	17

2.4.1.2	Motivation.....	18
2.4.1.3	Main Principles that make blind signature secure.....	19
2.4.1.4	Protocols under Blind Signature EVS.....	19
2.4.2	EVS based on Homomorphic Encryption.....	27
2.4.3	EVS based on Mixing Net.....	29
2.5	Cryptographic E voting Techniques.....	30
2.5.1	Public key encryption.....	32
2.5.2	RSA public key cryptosystem.....	32
2.5.3	Secret sharing.....	33
2.5.3.1	Threshold cryptosystem.....	33
2.5.4	Pseudo Random Number Generator.....	34
2.5.5	Cryptographic Hash Functions.....	35
2.5.6	Bulletin Board.....	36
2.5.7	OSCP protocol	36
2.5.8	Trapdoor commitment scheme.....	38
2.5.8.1	Trapdoor commitment scheme in E voting.....	38
2.5.9	Kerberos Authentication Protocol.....	39
2.5.9.1	Kerberos Authentication Protocol 5	39
2.5.10	Publicly Verifiable secret sharing PVSS.....	42
2.5.11	Elliptic Curve Cryptography.....	43
2.5.12	Public key Certificate.....	44
2.6	Related Work.....	45
Chapter Three: The proposed E voting Scheme.....		46
3.1	preparation stage.....	49
3.1.1	PVID Scheme.....	50
3.1.2	Kerberos Authentication protocol under E voting.....	61

3.2 Voting Phase	74
3.2.1 Ballot obtaining phase.....	74
3.2.2 Voting Submission Phase.....	77
3.3 Counting Phase	82
 Chapter Four: Analysis and Discussion.....	85
4.1 Security Requirements	85
4.1.1 A Method to Analyze Voting Systems.....	86
4.1.2 Formal Definitions of E voting Security Requirements.....	87
4.2 Simulation Result.....	102
4.2.1 Voting stage –Collude administrator problem	104
4.2.1.1 Discussion.....	104
4.2.2 Counting stage- Trapdoor commitment Scheme.....	106
4.2.2.1 Discussion	107
 Chapter 5: Conclusion and Future work.....	108
5.1 Conclusion.....	108
5.2 Future work.....	109
References.....	110
Appendices.....	119

List of Figures

Figure 2.1: Pre-voting Phase based on OASIS Standard	12
Figure 2.2: Voting Phase based on OASIS Standard	13
Figure 2.3: Post-voting Phase based on OASIS Standard	13
Figure 2.4: E voting general Requirements	15
Figure 2.5: A Conceptual View of blind signature Protocol in EVS	18
Figure 2.6: Evox-MA protocol	21
Figure 2.7: REVS voting protocol	23
Figure 2.8: REVS voting protocol Steps	24
Figure 2.9: Homomorphic encryption based voting protocols	28
Figure 2.10 : Mix-net based voting protocols	30
Figure 2.11: A Logical general flow of the Kerberos protocol	39
Figure 2.12: Kerberos Architecture supported different realm	40
Figure 3.1: A conceptual point of view for the proposed scheme	49
Figure 3.2: Pseudorandom number generation from a counter	51
Figure 3.3: ID-List Details	52
Figure 3.4: Blinding stage	53
Figure 3.5: Voter-PVID authority interaction	53
Figure 3.6: Signing Stage	54
Figure 3.7: Message passing in signing stage	56
Figure 3.8: PVID obtaining stage	57
Figure 3.9: Voter-PG interaction	58
Figure 3.10: ANSI X9.17 for pseudorandom number generator	58
Figure 3.11: PG –Voter interaction	60
Figure 3.12: Voter-AS interaction	62
Figure 3.13 : AS-Responder interaction	62
Figure 3.14: <i>KIS-OSCP</i> key Generation	63
Figure 3.15: D-KIS-OSCP phases	66
Figure 3.16 : AS-Voter interaction-1	66
Figure 3.17 : AS-Voter interaction-2	68
Figure 3.18 : Voter-AS interaction	69
Figure 3.19 : TGS-Voter interaction	70
Figure 3.20 : Voter-B-voting server interaction	72
Figure 3.21 : B-voting server –voter interaction	73
Figure 3.22 : Voter-EBG interaction 1	74
Figure 3.23 : Voter-EBG interaction 2	75
Figure 3.24 : Voter-EBG-KG interaction	76
Figure 3.25 : Voter-EBG interaction	77
Figure 3.26: Voter-Dealer-Administrators interaction for ballot signing	81
Figure 4.1: Dizzy simulator starting up	102

Figure 4.2: Dizzy simulator/PVSS	103
Figure 4.3: Dizzy Results/PVSS	104
Figure 4.4: Dizzy simulator /Trapdoor commitment scheme	106
Figure 4.5: Dizzy Results / Trapdoor commitment scheme	107

List of Tables

Table 2.1: E voting Types	11
Table 2.2: Protocols Evaluation	26
Table 2.3: Comparison between E voting Scheme	31
Table 2.4: RSA Algorithm	33
Table 3.1: Password generator algorithm Description	59
Table 3.2: Ballot eliminated the duplicate	83
Table 4.1: Privacy requirement details	88
Table 4.2: Eligibility requirement details	90
Table 4.3: Uniqueness requirement details	92
Table 4.4: Fairness requirement details	94
Table 4.5: Receipt freeness (Uncoercibility) requirement details	96
Table 4.6: Accuracy requirement details	99
Table 4.7: Individual Verifiability requirement details	101
Table 4.8: Proposed scheme simulation	105

Abbreviations

[A]	Voting process is canceled or rejected due to some errors or violation
AS	Authentication Server
CRL	Certificate Revocation List
CRT	Certificate Revocation Tree
DB	Data Base
DES	Data Encryption Standard
DRE	Direct Recording Electronic
DoS	Denial of Service
EBG	Electronic Ballot Generator
EDE	Encryption Decryption Encryption
Evox-MA	Electronic-vox Managed Administrator
EVS	Electronic Voting Scheme
FOO	Fujioka, Okamoto and Ohta
ID	Identification
KADM	Kerberos Administrator Server
KG	Key Generator
[O]	Voting process is correct and the voter can continue the operation
OASIS	Advancing Open Standards for the Information Society
OSCP	Online Status Certificate Protocol
OSCP-KIS	Online Status Certificate Protocol-Key Insulated Signature
PG	Password Generator
PKI	Public Key Infrastructure
PRNG	Pseudo Random Number Generator
PVID	Pseudo Voter Identity Scheme
PVSS	Public Verifiable Secret Sharing
REVS	Robust Electronic Voting System
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
SHA-1	Secure Hash Algorithm-1
TGS	Ticket Granting Server
VH	Vote Here, a verifiable E voting protocol

Abstract

Voting is one of the most important activities in a democratic society. In a traditional voting environment voting process sometimes becomes quite inconvenient due to the reluctance of certain voters to visit a polling booth to cast votes besides involving huge social and human resources. The development of computer networks and elaboration of cryptographic techniques facilitate the deployment of E voting. In this work the researcher propose a secure E voting scheme that is suitable for large scale voting over the Internet. The scheme depends on the last two E voting protocols based on the blind signature, Evox-MA and REVS; it encompasses three distinct phases- that of registration phase, voting phase and counting phase. Each phase applies some cryptographic technique, schemes or modified protocols to enhance some security aspects as a Kerberos authentication protocol, PVID scheme, responder certificate validation. The theoretical proof and simulation results show that the scheme satisfies all E voting security requirements. By applying a PVSS, the researcher get more stable results than REVS blind signature protocol which suggested using a different password for each administrator. Also the proposed scheme adds more security enhancements. First, by applying more than one scheme, the Kerberos authentication protocol (it has been modified by adding a new entity (responder) derived from the OSCP-KIS protocol to verify voter certificate validity), PVID scheme and the converted Ferguson E cash protocol the researcher guarantee that only authorized voter vote. Therefore, limit the DoS attack against attackers, so the counter buffer will never be filled with a garbage votes. Second, detecting the double voting issued by the voters, by applying more than one mechanism (the converted Ferguson E cash protocol to operate under E voting, trapdoor commitment scheme, and modified PVID scheme. Finally, allow a valid vote to be repeated if fault tolerance occurred by applying a trapdoor commitment scheme.

الملخص بالعربية:

إن عملية التصويت هي إحدى أهم الأنشطة في المجتمعات الديمقراطية. تكمن الصعوبة في عمليات التصويت التقليدي حيث يتعذر على بعض الناخبين المخولين بالإدلاء بأصواتهم بالوجود في مراكز الاقتراع في فتره الانتخابات بالإضافة إلى أن عمليات التصويت التقليدي تستهلك الكثير من الموارد الإجتماعيه والبشريه .

سهل التطور الحاصل في قطاع تكنولوجيا المعلومات والتطور الهائل في شبكات الحاسوب والتقدم في اليات التشفير والحمايه عمليه التصويت الإلكترونيه .

في هذا البحث تم إقتراح اليه جديده امنه للتصويت الإلكتروني والتي تناسب التصويت على نطاق واسع مثل الإنترنت. تعتمد الآليه الجديده على أحدث بروتوكلين من بروتوكلات التصويت الإلكتروني والتي تستند على اليه التوقيع الاعمى، وتشمل ثلاث مراحل رئيسيه مرحله التسجيل ومرحله التصويت ومرحله الفرز. كل مرحله من هذه المراحل تطبق مجموعه من الآليات والبروتوكلات الامنه لتحقيق بعض متطلبات الأمان والحمايه للتصويت الإلكتروني.

أثبتت النتائج النظرية أن الآليه الجديده امنه حيث حققت جميع متطلبات الأمان والحمايه للتصويت الإلكتروني كما أثبتت النتائج التحليلية أن الآليه الجديده استطاعت التغلب على بعض المشاكل الموجوده لدى REVS بفاعليه وإتزان أكثر، كمشكله التامر . عزز البحث بعض الجوانب الأمنية التي أهملها كل من البروتوكلين التي إستند البحث اليهما حيث:

- ✓ دمجت الآليه الجديده مجموعه من البروتوكلات والآليات لتضمن للناخبين المخولين فقط الإدلاء بأصواتهم الكترونيا وهذا بدوره قلل من DoS attack.
- ✓ لا يحق للناخب الإدلاء بصوته أكثر من مره ،حيث دمجت أكثر من اليه لمنع تكرار التصويت.
- ✓ في حاله وقوع خطأ في الشبكه أو تعذر على الناخب الاتصال بالإنترنت لسبب ما أثناء الإدلاء بصوته فقد سمحت الآليه الجديده للناخب بإدلاء صوته وممارسه حقه الانتخابي.

Chapter One

Introduction

1.1 Introduction

Elections allow people to choose their representatives and express their preferences for how they will be governed. Naturally, the integrity of the election process is fundamental to the integrity of democracy itself. The election system must be sufficiently strong to withstand a variety of fraudulent behaviors and must be sufficiently transparent and comprehensible that voters and candidates can accept the results of an election.

Nowadays, most governments start realizing the important for E voting as such a development will have many benefits towards elections and democracy itself; it will increase the number of voters, facilitate the casting of votes by voters from different places, not only from a particular polling station, which will help to reduce abstention rates, and accelerate vote counting and the delivery of voting results.

Specifically, the idea of E voting began in 1970 by the development of Direct-Recording Electronic (DRE) systems, and it had been used in voting stations in DRE cabinets, where the votes are stored electromagnetically (Kiayias, et. al, 2006). By 1990 when World Wide Web (WWW) became widespread there have been more attempts to apply E solutions to make democratic process easily. So E voting is applied again, in 1997 the idea was extended again when Monterey County, California experimented the first voting by mail system (Hirschberg, 1997).

Till now, the idea of E voting is applied more and more in many countries over the world and achieve a high degree of success as a Geneva pilot project (Cavadini & Cimasoni, 2007) indicated the increased number of participation with the introduction of E voting by a 20% over 8 years from an average of 30%-35% to an average of 50%-55%.

Also the united kingdom (St. Albans, Sheffield and Liverpool, 2006) test many E voting systems from 2000 to 2006 with a variant methods mainly internet (I-voting), with a high degree of success with some related problem related to the

difficulty of establishing a secure communication channel between voters and election server.

Despite the high percentage of success in E voting implementation projects, a few people have doubts about the privacy, security and accuracy of the election. They cannot easily trust the voting system unless the security of the system is greatly enhanced. Many controversies have been raised and many inconsistencies have been reported to be experienced with the real world electronic elections. The E voting experience in Ohio in 2004 is a well-known example; the incident caused much debate surrounding the evidence about vote miscount and modification.

In the recent two decades E voting became a hot research topic in advanced cryptography, posing several new challenges to fulfill voting general requirements. The challenge arises primarily from the needs to convince the voters that security and democracy requirements such as privacy, accuracy, receipt-freeness and verifiability were achieved and thus reduced their fear towards using E voting by providing them with a trusted E voting that they can rely on.

Many scientists and researchers (Chaum 1981, 1983; Fujioka, Okamoto & Ohta, 1992; Cohen & Fischer, 1985; Benaloh 1987; Cramer, Gennaro, Schoenmakers & Yung, 1996; Davenport, Newberger & Woodard, 1996; DuRette, 1999; Joaquin, Zúquete & Ferreira, 2002) explored in E voting cryptographic field in order to overcome the security issues in the election process. Each made his/her own contribution towards a trusted E voting but all agree about the major schemes that can be classified into three main categories: A blind signature scheme, the homomorphic encryption scheme and the mixing net scheme. Each of the above mentioned schemes underlies many protocols, these protocols try to achieve some general security requirements (e.g. by using a blind signature, the voter privacy will be guaranteed). Also, a combination between these schemes is possible depending on the requirements.

The protocols under blind signature scheme are considered as the most commonly implemented due to their practicality and applicability, at which the voter first obtains a token, which has been blindly signed by the administrator and which is only known to the voter her/himself. Later, the voter sends her vote anonymously, with this token as proof of eligibility to the auditing for counting.

While, in the homomorphic scheme the voter cooperates with the administrator in order to construct an encryption of his/her vote. Then, the administrator exploits homomorphic properties of encryption algorithm to compute the encrypted tally directly from the encrypted votes. For the mixing net scheme is the most common approach to achieving anonymity. The general concept of mix nets is based on permuting and shuffling the messages in order to hide the relation between the message and its sender. However, the details, as to the implementation of mixing protocols, change depending on configurations and arrangements of mix-nets.

This study will propose a new scheme that is based on Evox-MA (DuRette, 1999) and REVS (Joaquim, Zúquete & Ferreira, 2002) E voting based blind signature protocols, the two recent blind signature protocols. This scheme will overcome above the problems associated with these two protocols and provided a solution to them. It will solve the collude administrators problem, by implemented a publicly verifiable secret sharing based on a threshold signature (PVSS), guarantee voters Authentication, via applied a Kerberos protocol and eliminated the double voting produced by the voter. Up to now, no complete solution has been provided for such problem in neither theoretical nor practical domain.

1.2 Problem Statement

Voting is widely regarded as an effective means for people to express their opinions on a given topic. The schemes used in voting have been evolved from counting hands in early days to system that include paper, punch card, mechanical lever, and optical scan machines. The democratic elections that use voting machines have shown that the winning margins could be less than error margins of voting systems themselves, which make the election an error prone task.

Theoretically, the intension of the voters and the voting schemes can affect the voting results (e.g. the conventional paper-based voting scheme isn't convenient for distant voters that were live far from their home so may they lose their right in voting , thus the accuracy of voting result may decrease.

Therefore, increasing emphasis has been placed on developing E voting scheme capable of providing more efficient services than the conventional voting schemes. The rapid growth of computer network and the advent of internet technology facilitate such development.

The using of E voting has the potential to reduce or remove unwanted human errors that may appear in the traditional voting methods as a conventional paper based schemes. In addition to its reliability, E voting can handle multiple modalities (such as voice assistant for handicapped persons) and provide better scalability for large elections. Frankly speaking, E voting is considered as an excellent mechanism that doesn't require geographical proximity of the voters.

In the practical domain field, to transform to an E voting, there is a need for employment of a cryptographic technique to overcome above the security issues in the election process. Many schemes where developed which are mainly classified to: EVS under blind signature, EVS homomorphic encryption and EVS based on mixing net (reviewing the literature). Each of these schemes underlies many protocols.

If considering the protocols under E voting that based on the blind signature, specifically the last two, Evox-MA and REVS. The main observations are that:

- Both Evox-MA and REVS have the collude administrators problem that arises when one or more administrator collude to prevent authorized voters from voting or cooperate to send forge votes.
- Additionally, there is no complete solution to guarantee that only authorized voters vote.
- For the DoS attack, neither Evox-MA nor REVS can deal with the attacker from filling the counter buffer with garbage votes.
- To ensure all E voting requirements had been satisfied, there is a need to implement more additional services.

1.3 Research Objectives

The objectives of this study are as follows:

- Solving Collude Administrator problem: This problem can be solved by applying a Public Verifiable Secret Sharing (PVSS) based on a threshold signature in the newly proposed scheme, so it becomes difficult for two or more administrator to collude to alter voting results in the ballot signing phase.
- Guarantee that only authorized voter vote: This can be achieved by applying more than one scheme, the Kerberos authentication protocol (it has been modified by adding a new entity (responder) derived from the OSCP-KIS protocol to verify voter certificate validity), Pseudo Voter Identity (PVID) scheme and the converted Ferguson E cash protocol. This will help in filtering the counter buffer from unauthorized votes by ensuring that only authorized voters are permitted to vote. Therefore, limit the DoS attack against attackers so the counter buffer will never be filled with garbage votes.
- Allow a valid vote to be repeated if fault tolerance occurred by applying a trapdoor commitment scheme.
- Detecting the double voting issued by the voters, by applying more than one mechanism (the converted Ferguson E cash protocol to operated under E voting, trapdoor commitment scheme, and modified PVID scheme as the voter certificate obtain only once and PVID authority supply only one PVID for each eligible voter and doesn't make any sign for the blinded identities if the voter had been signed before. By this any attempt for double voting will be easily detected.
- Provide a non-repudiation service: Neither voter nor any entity participate or interact with voters can deny such interaction or communication by applying the Kerberos authentication protocol and the bulletin board.
- Preventing attacker from keeps track of a user password and compromise the voter password by relying on the ANSI X9.17 PRNG cyclic generated random number encryption infrastructure (Kelsey et al. 1997).

1.4 Motivation

In 1869 Thomas Edison received US patent 90,646 for an “electronic voting device.” He tried to sell his invention to the Massachusetts legislative bodies, unsuccessfully. A century later, we are once again attempting to apply electronic wizardry to expedite the democratic process (Rivest, 2000).

It seems as though everything is being automated by computers today. With the recent explosion of growth on the World Wide Web, the ability to communicate more information faster and cheaper is at our fingertips. We have email, electronic newspapers, and videoconferencing all leading the trend towards a paperless society.

Elections themselves have not remained completely static. Absentee ballots have long been common. This idea was extended in April, 1997, when Monterey County, California experimented with the first voting by mail (VBM) system (Harris, 1999). Additionally, Direct Recording Electronic (DRE) systems have been used in polling stations since the 1970s. In DRE booths, unlike their mechanical counterparts, the tallies are stored electromagnetically.

On the other hand, elections influence the democracy in a country directly. So it is highly important to ensure that elections carried out electronically are at least as secure and reliable as conventional elections are. Thanks to the recent advances in the field of cryptography we can bring all these trends together and create a secure E voting system.

1.5 Significance of the Research

Voting is usually recognized as one of the main characteristics of democracy. E voting is a very recent idea regarding voting. Many researchers (Chaum 1981, 1983, Fujioka, Okamoto & Ohta, 1992, Cohen & Fischer, 1985; Benaloh 1987, Cramer, Gennaro, Schoenmakers & Yung ,1996, Davenport, Newberger & Woodard ,1996, DuRette, 1999, Joaquin, Zúquete& Ferreira, 2002) gave attention for E voting over the last two decades. Up to now, many E voting protocols and scheme had been proposed, and both the security as well as the effectiveness has been improved.

The research proposes a new secure scheme based on Evox-MA and REVS E voting protocols that are based on the blind signature. The research brought a new scheme that solve the problems associated with these protocols and enhances some security measures by applying more than protocol or scheme or modified one to the proposed scheme. Mainly the significance can be summarized as the following points:

- Applying a PVSS based on a threshold signature will solve the collude administrators problem and its better than the double signing authentication that REVS proposed.
- For the first time the Kerberos authentication protocol (it has been modified by adding a new entity (responder) that is derived from the OSCP-KIS protocol to verify voter certificate validity) had been applied beside PVID scheme, the converted Ferguson E cash protocol and the bulletin board mechanism that guarantee that only authorized voter vote and help in filtering the counter buffer from unauthorized votes.
- Applying a trapdoor commitment scheme to detect a double voting and allow a valid vote to be repeated if fault tolerance occurred.
- More than one mechanism were combined to detect the double voting issued by the voters, the converted Ferguson E cash protocol to operated under E voting, trapdoor commitment scheme, and modified PVID scheme as the voter certificate obtain only once and PVID authority supply only one PVID for each eligible voter and doesn't make any sign for the blinded identities if the voter had been signed before. By this any attempt for double voting will be easily detected.

- An algorithm is used that depend on the ANSI X9.17 PRNG cyclic generated random number encryption infrastructure that prevents the attacker from keep track of a user password and compromise the voter password (Kelsey et al. 1997).

1.6 Thesis Organization

In addition to the introduction, there are four other chapters. *Chapter 2* describes E voting in many terms (what is E voting?, E voting Types, E voting phases depending on the OSAIS standard architecture and the general requirement that any E voting scheme or system should satisfied). Furthermore, it describes the E voting schemes a comparison between these schemes is provided. The main concentration is on the E voting protocols based on the blind signature, a general overview is provided for them. In detail, the researcher describe the latest two, Evox-MA and REVS E voting blind signature protocols as the proposed scheme depends upon them. Finally, some cryptographic techniques that the researcher refers upon then were described. The related work section includes a summary for the papers that research mainly depends upon them. *Chapter 3* proposes a secure E voting scheme based on a combination of Evox-MA and REVS E voting blind signature protocols, the newly scheme consist mainly of three stage, each stage describe in detail and more than one modified or exit scheme is applied at each stage to enhances the security measure for the proposed scheme. *Chapter 4* includes an analysis of the proposed scheme, theoretically proven that the new proposed scheme meet the general E voting requirements and dizzy simulator show result obtained from such implementation. Finally, *Chapter 5* draws conclusions and suggests future work in this research.

Chapter Two

Literature Review

In this chapter the researcher provides the essential information to understand E voting systems. The researcher starts by defined E voting and its type's. Then, present the usual phases of an E voting process, and describe the properties required for an E voting system. Finally, outline the main cryptographic primitives that are used in this thesis. The last section will be the related works that mention the main papers on which the research mainly depend on.

2.1 What is E voting?

Generally, the term “E voting” is used, in variety of different ways and it encompasses all voting techniques involving E voting equipment, including voting over the internet, using booths in polling stations and sometimes even counting of paper ballots. Specifically, E voting is any voting method where the voter’s intention is expressed or collected by electronic means. Based on the voting equipment and voting location, there are five types of E voting:

- **DRE voting** (Kohno et al.2004): Direct Recording Electronic (DRE) machine is physically an electronic equipment with running special purpose voting software. It lacks a tamper proof audit-trail. Satisfying accuracy and verifiability is almost impossible at DRE voting since any fraud during the voting process is unrecoverable and undetectable. This is similar to the current paper-based voting systems. The votes are cast inside a voting booth at a polling site; however, cast votes are recorded in electronic ballot boxes.
- **Poll-site voting** (Kohno et al.2004): In poll-site voting, the votes are cast by using public computers at a polling site. Voting booths are not used, but a public polling-site is provided. The computers at the site are connected over a closed and controlled network. Cast votes are recorded by a counting authority server instead of electronic ballot boxes. Voters can be authenticated and authorized at the site before allowed to access to the voting machines, or they can have some voting credentials prior to the voting period.

- **Poll-site kiosk voting** (Report of the National Workshop on Internet Voting, 2001): In poll-site kiosk voting, the votes are cast inside a voting booth at a polling site as in DRE voting. Typically, voting booths at the site contain electronic voting terminals, and they are connected with a closed and controlled network. Cast votes are recorded by a counting authority server instead of electronic ballot boxes. Voters are authenticated and authorized at the site before allowed access to the voting booths. Votes are cast using the terminal inside the voting booths.
- **Poll-site Internet voting** (Report of the National Workshop on Internet Voting, 2001): In this type, the votes are cast by using public computers at a polling site over Internet. Voting booths are not used, but a public polling-site is provided. The computers at the site are online over an uncontrolled network. Cast votes are recorded by a counting authority server instead of electronic ballot boxes. Voters can be authenticated and authorized at the site before allowed access to the voting machines, or they can have some voting credentials prior to the voting period.
- **Remote Internet voting:** Voters cast their votes over Internet. For authentication, the credentials of voters are verified prior to the voting period through the use of a password or some type of authentication token. As Table 2.1 holds a comparison between E voting types.

Table 2.1: E voting Types

	Stand-alone Voting	Networked voting	
		Controlled Network	Uncontrolled Network
Paper voting	Paper based voting	-----	-----
E voting	DRE voting	Poll-site kiosk voting	Poll-site Internet voting
		Poll-site voting	Remote Internet voting

2.2 E voting Phases

The basic process of any democratic election is almost standard although a wide variety of voting systems and protocols exist.

In general, this process consists of the following four tasks (Cranor & Cytron, 1997, p2; E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008). As shown in Figures 2.1 and 2.2.

- Pre-voting phase

This phase consists of two major parts according to OASIS standard (E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008):

(1) Candidate Nomination Process: for the candidate to be nominated, he/she should meet some legal restrictions according to the national legislative law (e.g. should be old enough), at last a nomination process result in candidate list that contain all nominated candidate.

(2) Voter Registration Process: at which all legitimate voters are registers depending on the local laws, voter should be above 18 years old. Finally, the result of this process is an election list that contains all legitimate voters.

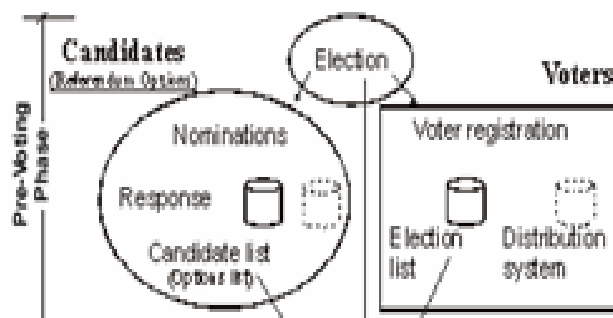


Figure 2.1: Pre-voting Phase based on OASIS Standard (E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008).

- Voting phase:

Depending on the result of the pre voting phase, in the voting phase each registered voter should authenticate his/herself as an eligible voter then the voter can cast his/her vote.

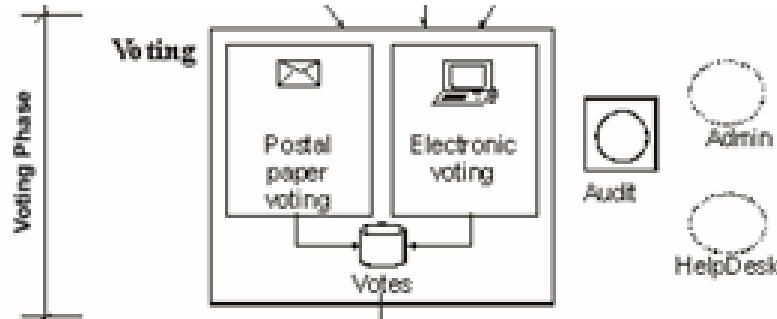


Figure2.2: Voting Phase based on OASIS Standard (E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008).

- Post Voting phase.

This phase consists of two major activities according to OASIS standard (E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008):

(1) Counting: The most critical step as it determine the list of election winners, the possibility of recounting should also be considered if either the input increased (votes) or in a case of multiple counters

(2) Result: After the counting is finished, the result of election will be available; it will be analyzed again by an auditing team and system administration

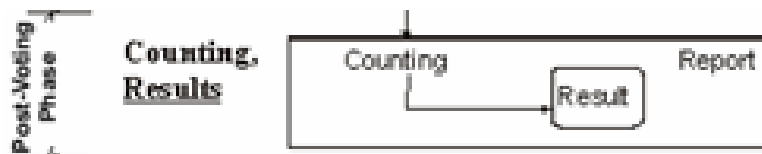


Figure2.3: Post-voting Phase based on OASIS Standard (E-vote: Election Markup Language 5.0 approved as OSAIS standard, 2008).

2.3 E voting General Requirements

There is a wide variety of e-voting requirement definitions (Cranor & Cytron, 1997, p2; Fujioka, Okamoto & Ohta, 1992; Mote, 2007; Cramer, Gennaro, Schoenmakers & Yung (1996); Cranor & Cytron, 1997, p3) with different naming convention such as requirements, properties, characteristics etc. These requirements can be grouped and summarized as follows; (see Figure 2.4):

- **Eligibility, Democracy (Authentication):** Only eligible and authorized voters can vote and each voter can vote only once.
- **Privacy:** All votes must be secret. No participant other than a voter should be able to determine the value of the vote cast by that voter, in other words, neither election authorities nor anyone else can link any ballot to the voter who cast it.
- **Receipt-Freeness (Uncoercibility):** No voter should be able to convince any other participant of his/her vote.
- **Fairness:** Nothing must affect the voting. No participant can gain any knowledge about the (partial) tally before the counting stage.
- **Accuracy:** The dishonest voter cannot disrupt the voting. No one can know the result of the voting. Every participant should be convinced that the election tally accurately represents the sum of the votes cast. It is not possible for a vote to be altered, it is not possible for a validated vote to be eliminated from the final tally, and it is not possible for an invalid vote to be counted in the final tally.
- **Individual Verifiability:** Each eligible voter can verify that his/her vote was really counted.
- **Universal Verifiability:** A system is verifiable if anyone can independently verify that all valid votes have been counted correctly. Any participant or passive observer can check that the published final tally is really the sum of the votes.

In addition to these requirements, (Cavadini & Cimasoni, 2007) proposed four extra properties that an E voting system should possess. Convenience and flexibility are the most important prosperities for ensuring a high voter turnout, something that is often desired but not always achieved.

- ✓ **Convenience:** A system is convenient if it allows voters to cast their votes quickly, in one session, and with minimal equipment or special skills.
- ✓ **Flexibility:** A system is flexible if it allows a variety of ballot question formats including open ended questions.
- ✓ **Mobility (Scalability):** A system is mobile if there are no restrictions (other than logistical ones) on the location from which a voter can cast a vote.
- ✓ **Robust:** All security requirements are fully satisfied, despite failure and/or malicious behavior by any (reasonably sized) coalition of parties (voters, authorities, outsiders).

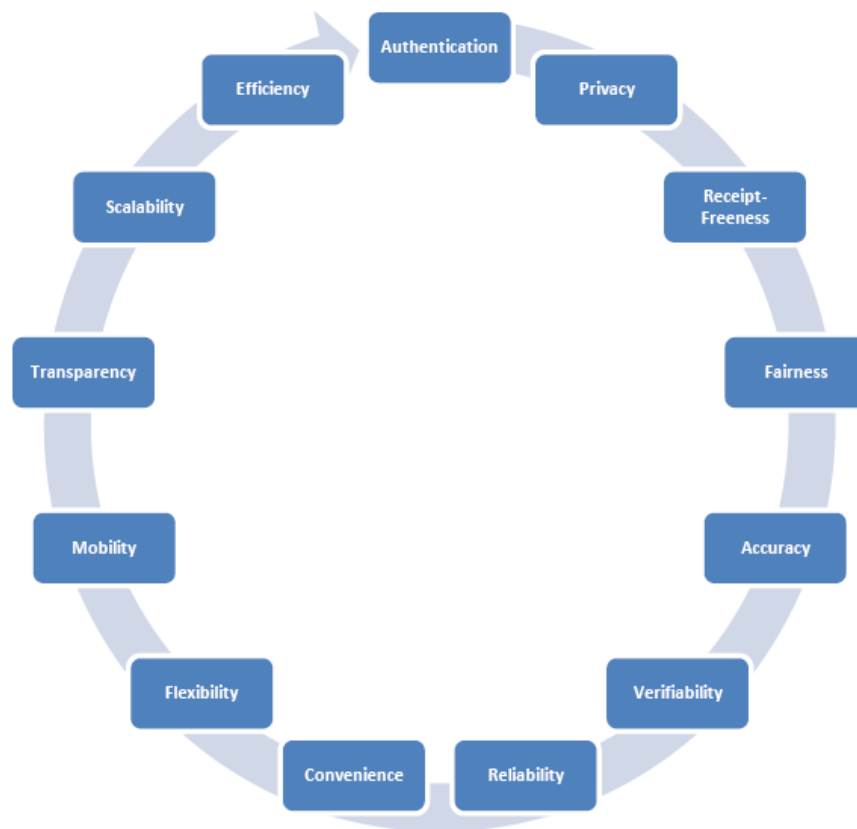


Figure 2.4: E voting Requirements

2.4 E voting Schemes

2.4.1 EVS Based on Blind Signature

The concept of blind signature was introduced by David Chaum (Chaum 1981, 1983). Chaum demonstrated the implementation based on RSA signatures. It allows the realization of secure voting schemes, protecting the voter privacy.

Initially the blind signature is used within E cash system (E cash) to guarantee owner anonymity, as in E voting scheme the motivation is to keep the voters anonymity as well, so this technique can be applied (Wen et. Al, 2009).

The idea of blind signature allows a signer to sign a document without revealing its contents similarly in a real life world to sign a carbon paper lined envelopes. Writing a signature on the outside of such envelope leaves a carbon copy of the signature on a slip of paper within the envelope. When the envelope is opened, the slip will show the carbon image of the signature.

A distinguishing feature of blind signatures is their unlinkability: The signer cannot derive the correspondence between the signing process and the signature, which is later made public.

The blind signatures can be accomplished by the following steps:

(1) The authority key is given:

(e, n) public key of the signer

(d, n) private key of the signer

(2) The voter's purpose is to let the authority to sign the vote, say v, without revealing its content (Blind Signature).

The voter generates a random number, r that satisfying the following formula

$$\gcd(n,r)=1 \dots\dots\dots(2.1)$$

The voter using this random variable r and authority public key component e to blind his/her vote and calculates

$$x = (r^e v) \bmod n. \dots\dots\dots(2.2)$$

(3) The voter asks the authority to sign the vote using its private key. Noted that the authority cannot derive any useful information from x.

$$t = x^d \bmod n \dots\dots\dots(2.3).$$

(4) The authority sends the signed vote to the voter.

$$\begin{aligned} t &= x^d \bmod n \\ t &= (r^e v)^d \bmod n \\ t &= (r^{ed} v^d) \bmod n \\ t &= r v^d \bmod n \dots\dots\dots(2.4) \end{aligned}$$

(5) As the voter know the random value r , she /he can remove it from the signed vote by taking r^{-1} to both side in (6)

$$\begin{aligned} r^{-1} t &= v^d \bmod n \\ s &= v^d \bmod n \dots\dots\dots(2.5) \end{aligned}$$

Where s is the vote v signed by the use of the authority private key preventing the authority from learning the signed vote v .

2.4.1.1 Implementation of blind signature Protocol in EVS

A blind signature protocol is similar to a digital signature except that it allows a person to get another person to sign a message without revealing the content of the message. In EVS, a ballot is blinded in order to achieve its confidentiality requirement .For simplicity, a protocol with two authorities; mainly a validator and a tailler are used to demonstrate how a blind signature is employed in EVS. A voter is required to get the signature of the validator when he votes. To ensure the secrecy of his/her ballot, a voter cast a ballot, B , blinds a vote using a random number and send it to the validator .Let (n,e) be validators public key and (n,d) be his/her private key. A voter generates a random number r such that $\gcd(r, n) = 1$ and sends the following to the validator $B' = (r^e B) \bmod n$.

The random number r conceals the ballot from the validator. The validator then signs the blinded ballot after verifying the voter, the signed value is $S' = (B')^d = (r^e B)^d \bmod n$.

After receiving the validated ballot, the voter unblinds the ballot, to get a true signature of a validator S by computing $S = S' r^{-1} \bmod n$.

The voter then sends his/her ballot together with validator signature to the tallier. The tallier verifies that if the ballot was correctly validated, then the ballot is valid. Figure 2.5 illustrate the employment of blind signature in EVS.

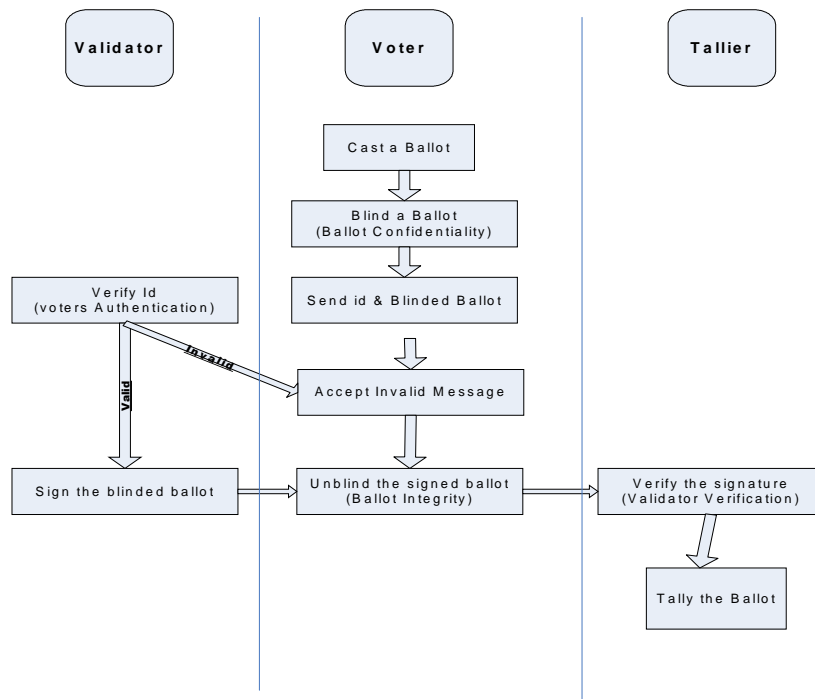


Figure 2.5: A Conceptual View of Blind Signature Protocol in EVS (Subariah et. al, 2003).

2.4.1.2 Motivation

Blind signatures may be useful in a voting protocol to perform the registration stage: where the registrar signs the ballot of a voter (after verifying the voter is eligible), without knowing its content. Then the signed ballot may be anonymously sent to the tallier who can then verify the signature and count the ballot.

Blind signature is very popular in practice due to their efficiency as communication and computation overhead is fairly small even when the number of voter is large. Also, blind signature and protocols underlying it can be easily managed and realize election with any type of voting (e.g. yes /no ,multiple candidates).

2.4.1.3 Main Principles that make blind signature secure

- A blind signature is secure if it can be proved that the identity of the holder of the signature is never revealed nor the content which is signed.
- Also the unconditional anonymity of the holder of the signature must be guaranteed even in the case of collusion this is known as a blindness property.
- Additionally, for a blind signature to be secure it must also be proven that blind signature can't be forged; even if a number of blind signatures are collected it must still be impossible for an attacker to forge the signature. Formally stated this means that if you have received j blind signature, its impossible to compute signature number $j+1$. This is also known as a non-foregability property.

2.4.1.4 E voting protocols based on blind signature

Protocols under blind signatures are very popular in practice due to their efficiency as communication and computation overhead is fairly small even when the number of voters is large and their support for any type of voting. These protocols can easily be managed and realize elections with multiple candidates.

However, the voter has to act in more rounds (registration, voting, counting, verifying, complaining), in other words, every eligible voter should not abstain after the registration phase, otherwise a corrupted validator can add extra votes on behalf of abstaining voters. Frankly speaking, these protocols only provide individual verifiability and no universal verifiability.

Many election protocols based on blind signatures have been proposed, (Chaum 1988; Fujioka, Okamoto & Ohta, 1992; Horster, Michels & Petersen, 1995; Juang, Lei & Yu, 1998; Okamoto 1997, Radwin 1995, DuRette 1999, REVS 2002).

Chaum (Chaum 1981, 1983) pioneered the notion of E voting and then several protocols were proposed. However, these earlier protocols suffer from providing most E voting properties (See Table 2.2 which hold a comparison among these protocols). Later; Chaum proposed a protocol based on the sender untraceable email system, which assumes that at least one mix is trust. It has large communication complexity at the registration phase. Ballot tallying authority can immediately open ballots upon

receiving them and therefore leaking intermediate results can effect the voting. Also, voter must reveal his/her vote to prove that it was not counted correctly which violates privacy concerns. Fairness and privacy are violated.

In 1989, Boyd proposed a protocol based on multiple key ciphers, it is considered more efficient than Chaum protocols. There is one administrator to carry out elections and issue valid voting slips to every potential voter exactly one. However, the voting authority can easily falsify the ballots. Furthermore, the voting authority can substitute spurious votes of its choice in the final tally. Knowledge of the intermediate results could distort further voting. Thus, it is not fair and not verifiable.

The first practicable protocol ensuring both the privacy and the fairness is of Fujioka et al. (Fujioka, Okamoto & Ohta, 1992). The proposed E voting protocol is capable of solving the fairness problem by using the bit commitment function. No one, including the voting authority, can know the intermediate result of the voting. Thus, it prohibits the fraud by either the voter or the authority. The voter has to participate in three rounds and he has to send two messages through anonymous channel. By this the number of rounds appears in the protocol of Juang et al. (Cohen & Fischer, 1985; Benaloh 1987) is reduced.

Juang et al. Cohen & Fischer (1997) proposal introduces scrutineers other than administrator. The protocol uses threshold cryptosystem to guarantee the fairness among the candidates campaign. It preserves privacy of the voter against the administrator and scrutineers. The protocol of Radwin (Cramer, Gennaro & Schoenmakers, 1997) mainly concentrates on tracing double-votes, the protocol is constructed on the idea of double spending, and Chaum (1992), most of the requirements are not fulfilled properly. There are also several implementations that have been piloted in small-scale elections: The SENSUS system was the first to be implemented (Cramer, Gennaro, Schoenmakers & Yung ,1996).

The Davenport et al system was used to conduct student governmental elections (Davenport, Newberger & Woodard ,1996). The EVOX system was used at MIT for undergraduate association elections (Cohen& Yung, 1986). DuRette (1999) improved EVOX system in order to eliminate single entities capable of corrupting the election in Evox Managed Administrators (Evex-MA). As an administrator's

signature is the base requirement to make a ballot valid, nothing prevents the administrators from creating and submitting forge valid ballots, also the administrator can prevent a voter from voting, refusing to sign his/her ballot, or allow several votes from the same voter, signing several times for the voter.

The idea explored by DuRette is to ensure democracy by sharing the power of the administrators among several servers. In DuRette proposal there are n administrators, and t signatures of them are required to make a ballot valid. There was also introduced the manager server that will sign the list of t signatures of the administrators to allow $t \leq n/2$, the protocol goes as shown in Figure 2.6:

- First the voter fills a ballot and commits to it using a random bit string. Then he blinds the committed ballot and sends it to $t \leq n$ administrators for signing as Figure 2.6, step 1 shows, in the Evox-MA case the ballot is obtained from the manager.
- Each administrator verifies independently if had not already been signed for the voter, and if not they sign the blinded ballot, update the voter record to an already voted state and return the signed blinded ballot to the voter as Figure 2.6, step 2 shows.

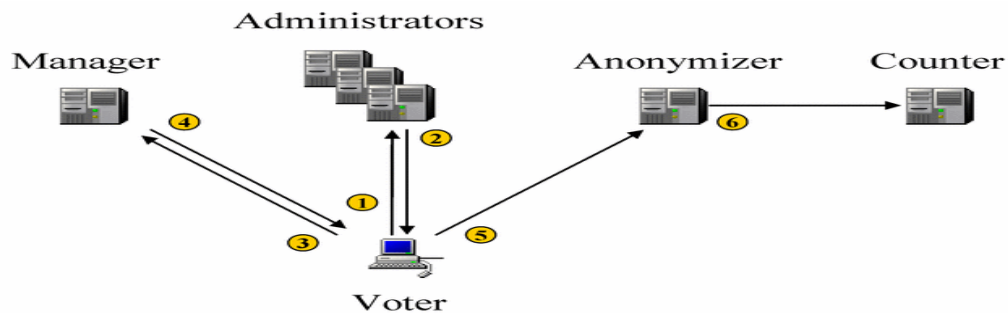


Figure 2.6: Evox-MA protocol (DuRette, 1999)

- After receiving all blinded signatures the voter removes the blinding layer and obtains a list of t signatures on the ballot. At this point the voter blinds the list of signatures and sends it to the manager as Figure 2.6, step 3 shows.
- If the manager had not already signed for the voter, he signs the blinded signatures list and returns it to the voter as Figure 2.6, step 4 shows.

- The voter receives the blinded signed signature list and unblinds it. Finally, to complete the voting process the voter encrypts the ballot, the bit commitment, the signatures and the signature on the list of signatures with the public key of the counter and he anonymously sends the encryption to the counter through the anonymizer as Figure 2.6, step 5 shows.
- When the election ends the anonymizer forwards the encrypted votes to the Counter in a random order as Figure 2.6, step 6 shows.
- The counter, after receiving the encrypted votes, decrypts them, removes the repeated votes and process the election tally.

In Evox-MA (DuRette, 1999) the democracy property is guaranteed by the administrators and the manager if they sign only once per voter. If $t \leq n/2$ the voter can get distinct lists of t signatures, therefore in this case is the manager that prevents a voter from obtaining more than one valid ballot. If $t > n/2$ the voter can only obtain of list of t signatures, therefore the manager is not needed to guarantee democracy. Apparently, in Evox-MA it is needed the collusion of t administrators and the manager to introduce valid votes.

However, Evox-MA does not offer the apparent collusion-resistance because it used only one password per voter for all administrators and also for the manager. None of these entities knows the password in advance, because a UNIX-like validation is used, i.e. the entity only has the digest of the password and not the real password. However, a small set of administrators, in collusion with the manager, can generate illicit valid votes using the voter's password once they get it. The fraud may work like this: x colluded administrators use the voter's password to get signatures from all the administrators not yet contacted by the voter.

Then they send to the manager a signed vote that he could accept and send to the counter. With n administrators and $n/2 + \Delta$ required signatures, x is equal to 2Δ . If, for improving performance, Δ is a low value (1 or 2), the possibility of attack is not negligible. If t is less than $n/2$, the manager itself can introduce votes without the participation of any other entity.

In Evox-MA there is some resistance to failures and collusion (DuRette, 1999). The voter must get t signatures from the administrators and one from the manager;

therefore the voter can lose up to $n - t$ signatures from the administrators, or tolerate the failure of them, without been prevented to vote. However if the voter loses the manager signature before submitting the vote, then he will be prevented to vote because the manager only signs once. A conclusion is drawn "the robustness of Evox-MA is higher than the one of Evox", but due to a weak authentication protocol it is not as good as it could be.

Both DuRette system and EVOX are very sensible to failures in communication or servers, these problems were solved by REVS which is proposed by Joaquim et al. as another implementation based on DuRette work. REVS based on DuRette work to make it more robust and scalable (Joaquim, Zúquete& Ferreira, 2002).

The flexibility of REVS architecture requires a flexible voting protocol (Joaquim, Zúquete& Ferreira, 2002). The only restriction made is to the number of required signatures to make a ballot valid, t , which must be greater than $n/2$, where n is the number of administrators. From the voters' point of view, the REVS protocol is divided in three steps as indicated in Figure 2.7.

(1) Ballot Distribution: The voter contacts a ballot distributor to get a blank ballot for a given election. The ballot distributor returns the requested ballot, the election's public key and the operational configuration of the election, all signed by the election commissioner. This is done in two phases. First the voter contacts a ballot distributor and provides a voter ID to receive the list of elections in which he can participate. Then the voter chooses the election and requests a ballot for it from a ballot distributor.

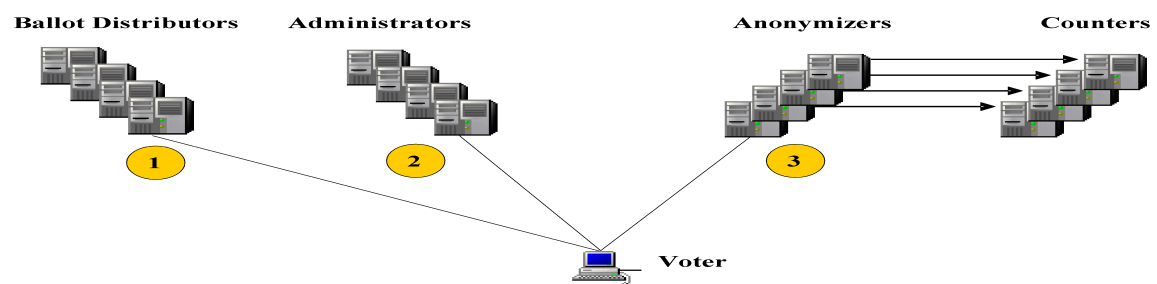


Figure 2.7: REVS voting protocol (Joaquim, Zúquete& Ferreira, 2002)

(2) Ballot Signing: After expressing his/her will on the ballot, the voter commits to the ballot digest with a random bit string and blinds the committed digest with a

random blinding factor. Then the voter sends the blinded committed digest of his/her vote to at least t administrators for signing. The administrator, after receiving a request for signing, verifies if it had already signed for the requesting voter. If not, he signs and saves the signature; if he had signed before, the administrator returns the previously saved data, i.e. the signature of the blinded committed ballot digest. After receiving a signature the voter updates it using an unblinding factor and verifies the correctness of the result using the original ballot digest and the administrator's public key.

This process is repeated until all required t signatures are collected. The voter's module can save the voter's answers, the bit commitment and the blinding factor into non-volatile storage, preferably provided by a mobile media, before using them. This enables the voter to stop and later resume its participation in the election, but can affect the voter's privacy because it can be used as a receipt as indicated in Figure 2.8.

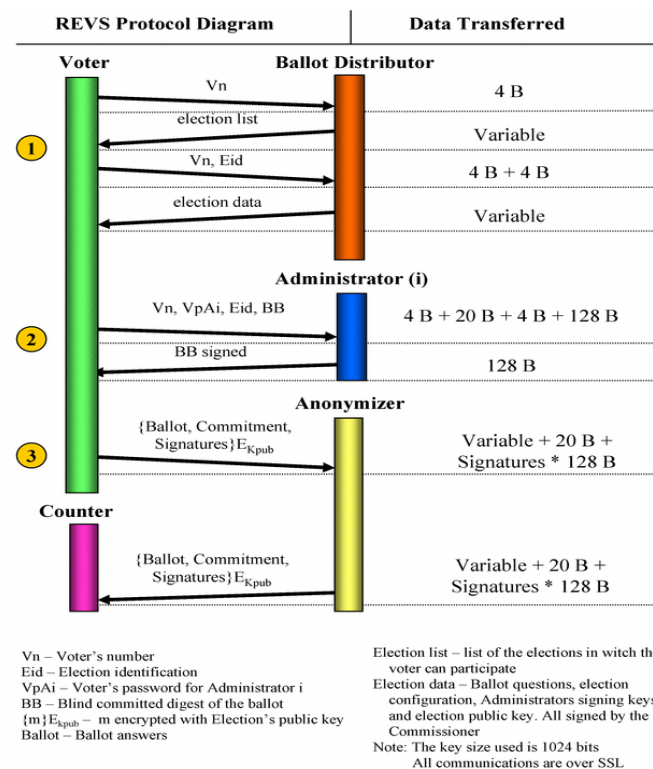


Figure 2.8: REVS voting protocol Steps (Joaquim, Zúquete& Ferreira, 2002)

(3) Ballot Submission: In this step the voter constructs the ballot submission package, joining the ballot, its signatures and the bit commitment. At this time the voter can

save this data into secure storage. Once again this is an optional step, because it helps improving accuracy but affects privacy. Then he submits this package, ciphered with a hybrid cryptosystem using a random symmetric session key and the election public key, through an anonymizer, concluding the voting protocol. The voter can submit the same package to any counter as many times as he feels necessary to be sure that the ballot reaches its destination. This means that different counters can get different sets of votes at the end of the election, and those sets may even contain repeated votes. A selected master counter obtains the final tally after gathering all the valid votes from the several counters and discarding the repeated ones. Any person with access to the ballots collected by all counters can act as a master counter. This fact increases the confidence in the election outcome. After collecting all votes the counting process involves the following steps:

1. Decipher the submission packages with the election's private key.
2. Verifying that all required signatures from administrators are present.
3. Removing repeated votes, which are the ones with the same bit commitment. If the length of the bit commitment is large enough (160 bits in REVS) the danger of collisions is negligible.
4. Tallying the remaining votes.
5. When using multiple counters, the master counter collects all previously verified votes. Then check for repeated votes using the bit commitment and proceeds with the final tally.

All the counters publish the contents of all received submission packages, and the administrators publish all the signatures provided for the blinded digests. After this publication the voter can verify if his/her vote was counted. If the vote is not present at the tally he can reclaim presenting, anonymously, the previously saved vote.

Table 2.2: Protocols Evaluation

	Nurmi (1991)	Fujiko (1992)	Davenport Et .al (1995/7)	Radwin (1995)	Cranor (1996)	Cramer Et al.7 (1997)	Du Rette (1999)
Double Voter Prevention	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ballot Confidentiality	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Universal Verifiability	Yes	Yes	Yes	Yes	Yes	Yes	Yes
No.of authority	2 Auth.	2 Auth.	2 Auth	1 Auth	3 Auth	n-Auth with same rule	n-Auth with diff Rule
Non Manipulability	No	No	No	No conclusion	No	No	No

2.4.2 EVS based on Homomorphic Encryption

Another commonly proposed way of achieving privacy in voting protocols is to use homomorphic encryption (Acquisti, 2004). A cryptosystem is said to be homomorphic when $E(s_1) \circ E(s_2) = E(s_1 \diamond s_2)$. Where E is a public encryption function, s is a secret function and \circ, \diamond is some binary operators. By homomorphic encryption it is possible to compute the combination of the individual messages without having to retrieve the individual messages themselves. So, the individual messages can remain confidential. The most two popular examples of homomorphic cryptosystems are ElGamal and Paillier cryptosystems.

Homomorphic encryption can be described in formal as follows. The probabilistic encryption function is $E_{pk} : R \times P \rightarrow C$. Where R is the randomness space, P is the plaintext space and C the cipher text space. The basic property of the encryption scheme is that $D_{sk}(E_{sk}(\cdot, x)) = x$ for all x . For homomorphic encryption, additionally assumed that the operations $\diamond, +, \circ$ define over the respective spaces P, R, C so that $\langle P, \diamond \rangle, \langle R, + \rangle$ are additive group and $\langle C, \circ \rangle$ is a multiplicative group. An encryption function E is homomorphic if, for all $r_1, r_2 \in R$ and all $x_1, x_2 \in P$ is holds that: $E_{pk}(r_1, x_1) \circ E_{pk}(r_2, x_2) = E_{pk}(r_1 + r_2, x_1 \diamond x_2)$.

The voting protocols based on homomorphic encryption, as the encrypted votes gather; result in the accumulation of votes. The voting result is then obtained from the accumulation of votes while no individual ballot is opened and the corresponding individual vote remains secret. Figure 2.9 displays an overall view of homomorphic encryption based voting protocols.

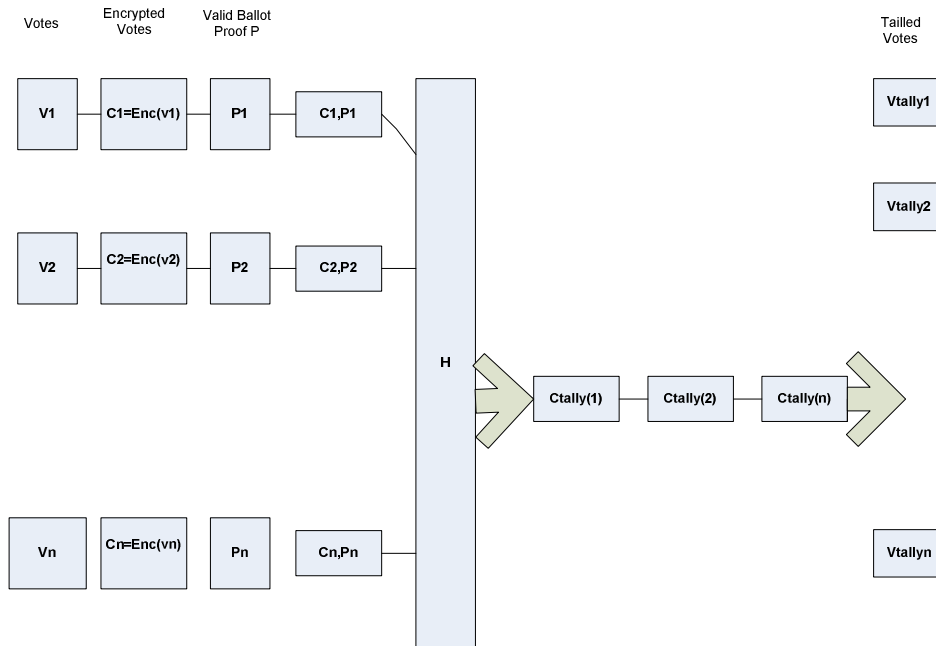


Figure 2.9: Homomorphic encryption based voting protocols (Hirt M. & sako K., 2000).

In homomorphic encryption based protocols (Benaloh & Tuinstra, 1994; Benaloh & Franklin, 2001; Acquisti, 2004; Hirt & Sako, 2000; Sako & Kilian, 1994; Benaloh 1987) voting results are obtained easily so ballot tabulations are conducted more efficiently when the number of candidates or choices is small. However, homomorphic voting has a drawback where each vote must be verified to be valid since correctness of the tallying cannot be guaranteed without validation. When the number of candidates or choices is large, computational and communicational cost for the proof and verification of vote validity is so large that homomorphic voting actually becomes inefficient for large scale elections. A great advantage of this approach is that voters may openly authenticate themselves to the voting servers; there is no need for anonymous channels to ensure voter privacy.

Electronic voting protocols based on homomorphic encryption have more security properties than other protocols, but their communication complexity is quite high (Hirt M. & sako K., 2000, Acquisti, 2004). They are most suitable for yes-no or 1-out-of-L voting. A known implementation of this approach can be found in a European Union project; the CyberVote project, funded by the European Commission, has developed a prototype system (Camenisch & Lysyanskaya, 2005).

2.4.3 EVS based on Mixing Net

Mix-networks (Mix-Nets) are the most common approach to achieving anonymity. It is based on permuting and shuffling the messages in order to hide the relation between the message and its sender. However, the details, as to the implementation of mixing protocols, change depending on configurations and arrangements of mix-nets.

A mix-net typically consists of a set of mix servers which are responsible for mixing the incoming inputs and producing a shuffled output. In mix-nets, there are n mix-servers M_1, M_2, \dots, M_n ; each with its own public key E_i and private key D_i . Each server processes the input messages. The process can be either re-encryption or decryption depending on the mix-net types. Then, each server permutes the processed messages and forwards them to the next mix server.

The first mix-nets are decryption mix-nets (Chaum 1981; Park & Itoh, 1993; Jakobsson & Juels, 2001) where messages are wrapped in several layers of encryption and then are routed through mix servers, each of which peel off a layer of encryption and then forward them in random order to the next one. In decryption mix-nets, decryption in each mix server is repeated until all layers are removed. One of the well-known implementation of decryption mix-nets was Onion routing (Camenisch & Lysyanskaya, 2005; Goldschalg, Reed & Syverson, 1999).

Later, re-encryption mix-nets were introduced (Sako & Kilian, 1995; Golle & Jakobsson, 2004; Jakobsson, Jules & Rivest, 2002) where the incoming messages are not decrypted, but re-encrypted in each mix server. In re-encryption mix-nets, decryption occurs after shuffling is completed. The major drawback of the decryption and re-encryption mix-nets is that one server may compromise and cheat by removing or replacing any number of items. Therefore, they are extended to be verifiable. In verifiable mix-nets, a mix server additionally has to prove in zero knowledge that it decrypts/re-encrypts and shuffles the inputs correctly.

There are several approaches to obtaining verifiable mix-nets; the main difficulty in these approaches is inefficiency of proof techniques (Pfitzmann 1994; Abe, 1998; Furukawa & Sako, 2001; Neff 2001). The call for proving that the mixing is correct causes an excessive computational cost for mix servers, so their

implementation is not practical. Using mix-nets in voting protocols is generally called as mix voting. As a general approach, a voter casts his/her vote over a mix-net, and it is assumed that a vote cannot be linked to a particular voter.

In mix-net based voting protocols, voters prepare their ballots stating for whom they wish to vote and encrypt their ballots. Then, they send their cast mixes to them in a random order. Later, it re-encrypts/decrypts the votes and forwards all votes to the next mix server. The next mix server takes the votes and shuffles them in the same way as the first server. Successively, each mix server takes the votes sent by the previous server, shuffles them and sends the produced list to the next mix server. The list produced by the last mix server is called the final votes list. The list is counted after the final decryption/encryption and published. Figure 2.10 shows a general view of mix-net based voting protocols.

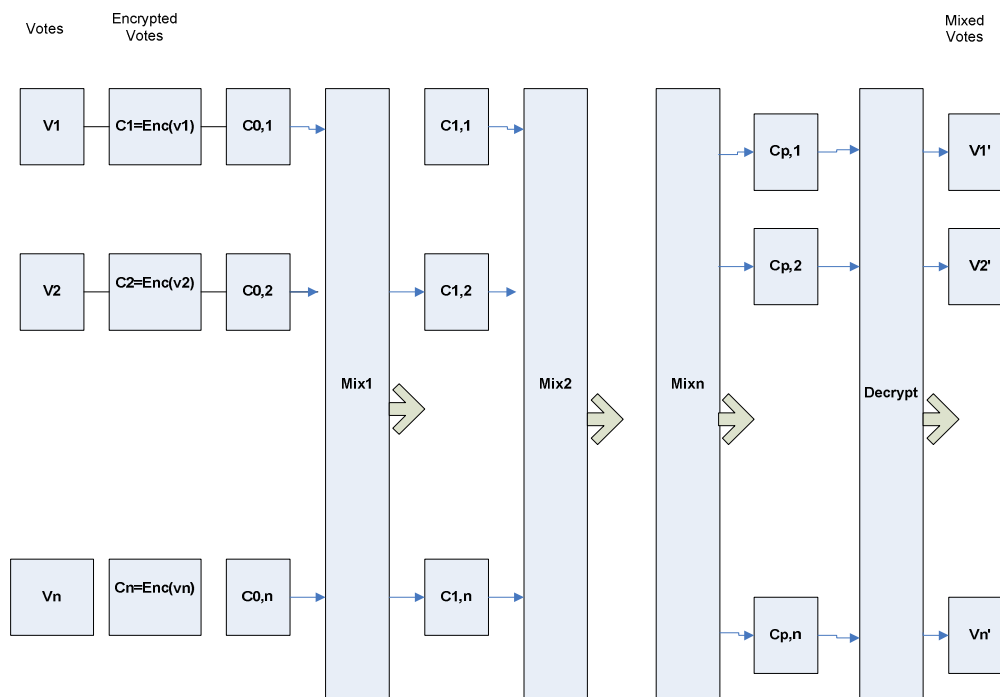


Figure 2.10: Mix-net based voting protocols (Jakobsson M., & Juels A., 2001)

Some of the protocols in this type have different implementations. Vote Here VHTi is a commercial implementation which focuses on voter-verifiability. Sure Vote is an enhancement of the mix-net approach by Chaum, which incorporates a voter verifiable component and uses proprietary printing equipment (Chaum 2004). (See Table 2.3, which holds a comparison between these major schemes).

Table 2.3: Comparison between E voting Scheme

	Homomorphic Based Encryption scheme	Blind Signature scheme	Mix-Net scheme
Mathematical Structure	Much	Little	Medium
Tallying	Decryption of the totals	Decryption of individual votes	Decryption of individual votes
• Cost			
✓ Voting	high	small	medium
✓ Tallying	small	Very small	medium
✓ Verification	small	Local only	medium
Assumption and restriction	Restrict ballot format	Anonymous channels or private voting booths	Some ballot restriction
Scalability	Medium	Good	medium

2.5 Cryptographic E voting Techniques

This will give a simple and clear idea of the basic concepts and techniques behind the design and implementation of many E voting protocols.

2.5.1 Public key encryption

In the public key encryption, also known as asymmetric encryption, there are two keys: an encryption key K_{pub} (public key) and a decryption key K_{pri} (private key). The encryption of a message m with K_{pub} results in c , to recover m from c using K_{pri} , as follows:

$$c = E(K_{pub}(m))$$
$$m = DK_{pri}(c) = DK_{pri}(EK_{pub}(m))$$

In E voting a public key cryptosystem is normally used to provide secure authentication to the voters, or to establish secure connections between the voters and the electoral servers.

2.5.2 RSA public key cryptosystem

The most known and used algorithm for public key encryption is the RSA, proposed by (Rivest, Shamir & Adleman, 1977). The security of the RSA algorithm is based on the problems of factorization and calculation of modular logarithm for large numbers. In E voting the use of the RSA, or some derived algorithms is common on blind signature based voting systems. It is also used in the construction some of mix-nets. The details of the algorithm are shown in Table 2.4

Table 2.4: RSA Algorithm

Secret Values	p,q	Secret distinct large primes ,also calculate $\phi = (p-1)(q-1)$
Public value	n	$n=p.q$
Public key	e	$1 < e < n$, such that $\gcd(e, \phi) = 1$
Private key	p,q,d	$d < n$ such that $1 = e.d \bmod \phi$
Encryption	$c = m^e \bmod n$	
Decryption	$m = c^d \bmod n$	

2.5.3 Secret sharing

Secret sharing, as the name suggests, is called to the process of sharing a secret S among N parties so that only t or more parties can later recreate the secret. Each party P_i keeps his/her share s_i secret, so that just $m \geq t$ parties can recreate the secret S . Such a scheme it's called (t, N) threshold secret sharing scheme. The interest of this scheme is to prevent the ability of less than t parties to reveal the shared secret.

2.5.3.1 Threshold cryptosystem

In a threshold cryptosystem the secret sharing technique is used to share a private key K_{pri} among N parties, in such a way that at least t parties must cooperate to decrypt $E_{K_{\text{pub}}}(m)$, where m is an arbitrary message. These systems are called (t, N) threshold cryptosystems. Threshold cryptosystems usually include two algorithms (Desmedt 1993; Baek J. & Zhen Y., 2004; Libert B. and Quisquater J, 2003):

✓ Key Generation protocol:

All the N parties are involved in the generation of the share public key K_{pub} . At the end each one receives its share of the private key K_{pri} .

✓ Verifiable Decryption protocol: Allows t parties to cooperatively decrypt an encrypted message $E_{K_{\text{pub}}}(m)$ in a way that everyone can verify that the decryption was performed correctly. This process should not give anyone the ability to decrypt alone any other messages encrypted with the same public key. In some E

voting protocols there is an election public key, used to encrypt the ballots. The use of a threshold cryptosystem for the election's private key brings obvious improvements to the system security, because votes cannot be revealed without the cooperation of t election authorities.

2.5.4 Pseudo Random Number Generator

Pseudo random number generator (PRNG) (Kelsey et al. 1997) is a deterministic algorithm to generate a sequence of numbers with little or no discernible pattern in the numbers. The sequence is not truly random since it is determined solely by a relatively small set of initial values. Although sequences that are closer to truly random ones can be generated using hardware random number generators, most pseudo random generator algorithms produce sequences which are uniformly distributed. Getting truly random data is typically based on nondeterministic physical phenomena. In the deterministic environment of computer systems, people often use deterministically generated pseudorandom data. The truly random data are used only for deterministic pseudorandom number generators and after seeding, an arbitrary amount of pseudorandom data is always available. The PRNG is in fact a deterministic finite state machine, which implies that it is at any point of time in a certain internal state.

This PRNG state is kept confidential since the PRNG output must be unpredictable. Many classes of PRNGs exist, but the goal of a PRNG in cryptography is the production of pseudo random data that are computationally indistinguishable from statistically ideal random data. A PRNG is cryptographically secure, on condition that it is computationally infeasible to predict the next output even if all the previous outputs and the complete algorithm are given. Basic types of PRNGs utilize linear feedback shift registers, NP hard problems of number and complexity theory and typical cryptographic functions/primitives. Mechanisms necessary for recovering from the state compromise are used only in the last category.

2.5.5 Cryptographic Hash Functions

A cryptographic hash function is a hash function h with certain additional security properties, which takes an arbitrary size input x and outputs a fixed length output $h(x)$.

Although a cryptographic hash function is deterministic and efficiently computable, it should behave as much as possible like a random function. Hash functions are assumed to be public; therefore if x is given, anyone can compute $h(x)$. Digital signatures and data integrity are the most common cryptographic uses of hash functions. With digital signatures, a long message is usually hashed (using a publicly available hash function), and only the hash-value is signed. The party receiving the message then hashes the received message and verifies that received signature is correct for this hash-value. This saves both time and space compared to signing the message directly. In order to meet the requirements of a signature scheme the following three properties are required of a cryptographic hash function h :

- ✓ *Pre-image resistance* means that given $h(x)$, it is computationally infeasible to extract any bits of x .
- ✓ *Second pre-image resistance* means that given x , it is computationally infeasible to find y such that $h(x) = h(y)$.
- ✓ *Collision resistance* means that it is computationally infeasible to find any x and y such that $h(x) = h(y)$.

MD5 (Rivest, 1994), SHA-1(RFC 3147,2001), SHA-256(RFC 4868,2007) are well known hash algorithms. The MD5 algorithm produces a 128-bit message digest used to validate data integrity. The SHA-1 algorithm produces a 160-bit message digest and is therefore considered a stronger algorithm than MD5 (Rivest, 1994).

SHA-1(RFC 3147, 2001) is utilized in a broad range of popular security applications and protocols. The SHA-256 hashing algorithm extends the size of the digest to 256 bits for heightened security. Wang *et al.* showed the collisions for MD5.

2.5.6 Bulletin Board

Some cryptographic protocols, to prove their correctness, need a space where everyone can write and read but not delete information. A Bulletin Board (BB) is a public broadcast channel with universally accessible memory where a party may write information via secure communication in the designated areas. The information can be read by any party. Bulletin boards are commonly used in E voting protocols. All communications with the bulletin boards are public and therefore can be monitored. Generally, data already written into a bulletin board cannot be altered or deleted in any way, but it can be read or appended.

2.5.7 OSCP protocol

In the *Public Key Infrastructure* (PKI), a certificate is used to bind an entity's identity information with the corresponding public key. Nevertheless, certificates are revoked in case of breaking that binding before its expiration date. Thus, the certificate verifier must check not only the expiration date on the certificate but also the revocation information of it. A certificate revocation system can be implemented in several ways. The most well-known method is to periodically publish a *Certificate Revocation List* (CRL) (Housley & Polk, 2002), which is a digitally signed list of revoked certificates and usually issued by the Certification Authority (CA). The main advantage of the CRL systems is its simplicity, but several problems are pointed out (Arens et al., 2000). Especially, the main disadvantage of the CRL systems is its high communication costs between the user and the repository, because the size of CRL will be quite long if the CA has many clients.

To overcome the shortcomings of the CRL, several revocation methods are suggested as follows. The Delta CRLs are issued more frequently and only include updates to the complete revocation list called Base CRL under (ITU/ISO Recommendation). CRL distribution points were specified in ITU/ISO Recommendation. CRL distribution points allow revocation information within a single domain to be divided into the multiple CRLs. So the CRL of each domain can be smaller than the full CRL. The *Certificate Revocation Tree* (CRT) was proposed by Kocher (1998). CRTs are based on (Merkle, 1990) Hash tree, in which the tree itself represents all certificate revocation information.

The status of any given certificate can be proved by using the path from the root to the target leaf. Therefore, the communication costs between the user and repository can be lower than those of CRL systems (Naor & Nissim, 1990) proposed the authenticated directory which improves the reduction in communication cost by balancing the hash tree. They introduced using a 2-3 tree, in which every node has two or three children. (Kikuchi et al., 2001) the binary hash tree is extended to k -ary hash tree in which any node has at most k children. Micali proposed the revocation system using hash chains. (Micali, 2002) taking into account both user's and CA's efficiency. The advantage of Micali's system is that the communication costs are very efficient, because the user may just obtain 160-bit hash value. It is necessary to obtain timely information regarding the revocation status of a certificate. The most popular mechanism that provides real-time status of a certificate is the *Online Certificate Status Protocol* (OCSP) by (Myers et al., 1999).

Online certificate status protocol is considered as a mean to check the validity of a certificate. If timeliness status information is required, OCSP is preferred. When AS request status information for wanted certificate to OCSP responder, the responder examines the status of the requested certificate and then returns a response including OCSP responder's digital signature for the response message (Malpani, Housley, and Freeman, 2003). At this moment, the status of the response is one of good, revoked or unknown, when voter receives the response message, voter first verifies the responder's signature and then accepts the response.

Generally, OCSP responder is a single server, and digital signature is a computation consuming operation, so if much verification is converged into the one responder single-point-of-failure problem or DoS is possible because of the heavy burden of all response processing. For OCSP to operate in a distributed E voting environment, it will consist of multiple responders and each responder shares the burden of OCSP response. Voters can select one of those responders and each responder will return the response including its digital signature.

To verify the responder's signature, AS must obtain the replying responder's certificate and check the status of the responder certificate again. Therefore efficient key management is required for multiple responders.

2.5.8 Trapdoor commitment scheme

A trapdoor commitment scheme (Chen, et. al, 2011) is a function with associated a pair of matching public and private keys. The main property that want from such a function is collision-resistance: unless one knows the trapdoor, it is infeasible to find two inputs that map to the same value. On the other hand, knowledge of the trapdoor suffices to find collisions easily. The trap door commitment scheme (Bresson, Catalano & Pointcheval (2003) is based on bit commitment scheme cryptosystem. A trapdoor commitment scheme consists of key generation algorithm, commitment function, and collision-finding function.

- Key Generation

The key generation algorithm, on input a security parameter l produces a modulus N product of two safe primes of size $l / 2$ together with a square h of maximal order in G . The public key is given by N and h . The factorization of the modulus is the private key (p, q) .

- Committing a Message

To commit to a message $m \in \mathbb{Z}_N$ the sender chooses a random number $r \in_R \mathbb{Z}_{N\lambda(N)/2}$ and sets $B = C(r, m) = h^r (1 + mN) \bmod N^2$, and sent (B, r, m) to the receiver.

- Collision-Finding Function

Now given a commitment $B = C(r, m) \in G$ together with the corresponding (r, m) , knowing the factorization of the modulus, one can find collisions, for any message m' as follows $r' = r + (m - m')d\lambda(N) \bmod N\lambda(N)/2$. Thus the receiver can get $B = C(r, m) = C(r', m') \in G$.

2.5.8.1 Trapdoor commitment scheme in E voting

Trap-door bit commitments were introduced in voting schemes as a means of solving the problem of coercion. As well as the convenience for the voters is an important property, schemes using bit commitments do not seem practical for use in large scale elections. In a trap-door bit commitment scheme, where a voter v has committed to a message M , it is possible for v to open M in many different ways. This may seem to contradict the purpose of commitment schemes, but the following scenario shows how this property can be useful in E voting:

1. A voter v commits to a voting intention B . v then provides the authority A with the information necessary to open the commitment on B , but keeps a secret value, the trap-door, to him. This enables only the voter to open the commitment in different ways.
2. In the tallying phase, A opens the commitment on B . No interaction from the voter is required.
3. If a coercer forces v to demonstrate how he has voted, v can use the secret trap-door to claim a voting intention different from his/her actual intention, without the coercer being able to detect it. But the main difficult requirement to achieve within a trap door commitment scheme is the secret keeping for the trap door value as one generally assumes the coercer has access to the same information as the voter.

2.5.9 Kerberos Authentication protocol

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography, a logical general flow for the Kerberos protocol is shown in Figure 2.11.

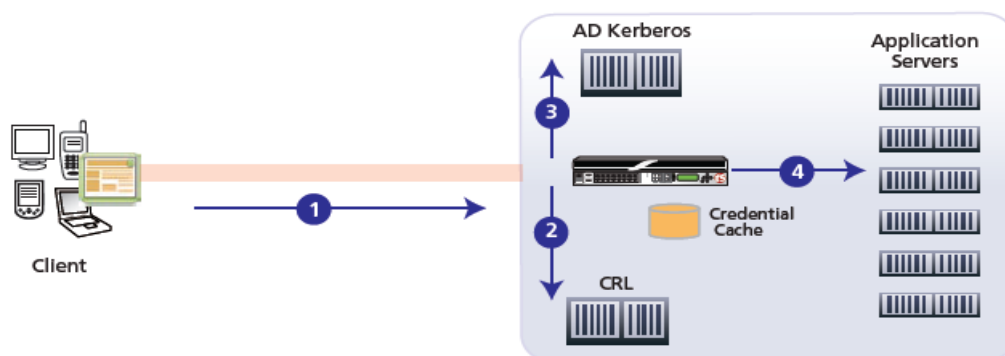


Figure 2.11: A Logical general flow of the Kerberos protocol

Due to the assumption by some researcher "The Internet is an insecure place". Many of the protocols used in the internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, other client/server applications rely on the client program to be "honest" about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server.

Some sites attempt to use firewalls (Scarfone K. & Hoffman P., 2009) to solve their network security problems. Unfortunately, firewalls assume that "the bad guys" are on the outside, which is often a very bad assumption. Most of the really damaging incidents of computer crime are carried out by insiders. Firewalls also have a significant disadvantage in that they restrict how your users can use the Internet. In many places, these restrictions are simply unrealistic and unacceptable.

Kerberos was created by MIT (Migeon, 2008) as a solution to these network security problems. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business; the business that will be utilized here is the voting over the internet (E voting).

In summary, Kerberos is a solution to the network security problems. It provides the tools of authentication and strong cryptography over the network to help you secure your information systems across your entire enterprise within a mutual authentication under the assumption that the underlying internet infrastructure is insecure. Kerberos has been invaluable to the E voting proposed scheme.

2.5.9.1 The Kerberos Authentication protocol version 5

Kerberos version 5 that specified in RFC 1510, which supported the different realm¹ architecture as Figure 2.12 shows.

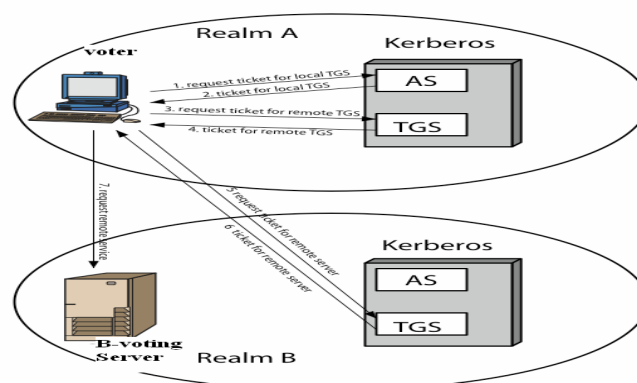


Figure 2.12: Kerberos Architecture supported different realm (Migeon, 2008)

¹ A Kerberos realm is a set of managed nodes that share the same Kerberos database, and are part of the same administrative domain

It consists of several sub-protocols (or exchanges). There are two basic methods by which a client can ask a Kerberos server for credentials. In the first approach, the client sends a clear text request for a ticket for the desired server to the AS. The reply is sent encrypted in the client's secret key. Usually this request is for a ticket-granting ticket (TGT), which can later be used with the ticket-granting server (TGS). In the second method, the client sends a request to the TGS. The client uses the TGT to authenticate itself to the TGS in the same manner as if it were contacting any other application server that requires Kerberos authentication. The reply is encrypted in the session key from the TGT. Though the protocol specification describes the AS and the TGS as separate servers, in practice they are implemented as different protocol entry points within a single Kerberos server.

Once obtained, credentials may be used to verify the identity of the principals in a transaction, to ensure the integrity of messages exchanged between them, or to preserve privacy of the messages. The application is free to choose whatever protection may be necessary.

To verify the identities of the principals in a transaction, the client transmits the ticket to the application server. Because the ticket is sent "in the clear" (parts of it are encrypted, but this encryption doesn't thwart replay) and might be intercepted and reused by an attacker, additional information is sent to prove that the message originated with the principal to whom the ticket was issued. This information (called the authenticator) is encrypted in the session key and includes a timestamp.

The timestamp proves that the message was recently generated and is not a replay. Encrypting the authenticator in the session key proves that it was generated by a party possessing the session key. Since no one except the requesting principal and the server know the session key (it is never sent over the network in the clear), this guarantees the identity of the client.

The integrity of the messages exchanged between principals can also be guaranteed by using the session key (passed in the ticket and contained in the credentials). This approach provides detection of both replay attacks and message stream modification attacks. It is accomplished by generating and transmitting a collision-proof checksum (elsewhere called a hash or digest function) of the client's message, keyed with the session key. Privacy and integrity of the messages

exchanged between principals can be secured by encrypting the data to be passed by using the session key contained in the ticket or the sub-session key found in the authenticator.

The authentication exchanges mentioned above require read-only access to the Kerberos database. Sometimes, however, the entries in the database must be modified, such as when adding new principals or changing a principal's key. This is done using a protocol between a client and a third Kerberos server, the Kerberos Administration Server (KADM). There is also a protocol for maintaining multiple copies of the Kerberos database

2.5.10 Public verifiable secret sharing PVSS

Secret sharing and its many variations form an important primitive in cryptography. The basic model for secret sharing distinguishes at least two protocols:

- 1) A distribution protocol in which the secret is distributed by a dealer among the participants.
- 2) A reconstruction protocol in which the secret is recovered by pooling the shares of a qualified subset of the participants. Basic schemes (Shamir 1979) for threshold secret sharing) solve the problem for the case that all players in the scheme are honest.

In verifiable secret sharing (VSS) the object is to resist malicious players (Stadler, 1996), such as:

- (1) A dealer sending incorrect shares to some or all of the participants
- (2) Participants submitting incorrect shares during the reconstruction protocol. In publicly verifiable secret sharing (PVSS), it is an explicit goal that not just the participants can verify their own shares, but that anybody can verify that the participants received correct shares.

2.5.11 Elliptic Curve Cryptography

Elliptical curve cryptography (ECC) is a public key encryption technique based on *elliptic curve theory* that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large primes. The technology can be used in conjunction with most public key encryption methods, such as RSA, and Diffie-Hellman. ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve. Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications. ECC was developed by Certicom (Menezes, 1993) a mobile E business security provider, and was recently licensed by Hifn, a manufacturer of integrated circuitry (IC) and network security products. RSA has been developing its own version of ECC.

ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptography purposes: they are relatively easy to perform, and extremely difficult to reverse.

2.5.12 Public key certificate

In cryptography, a public key certificate (also known as a digital certificate or identity certificate) is an electronic document which uses a digital signature to bind a public key with an identity, information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual. In a typical public key infrastructure (PKI) the signature will be of a certificate authority (CA). Here a PVID authority acts as a CA. In a web of trust scheme, the signature is of either the user (a self-signed certificate) or other users ("endorsements"). In either case, the signatures on a certificate are attestations by the certificate signer that the identity information and the public key belong together.

The operating principle of electronic certificates is based on encryption of information and trust. Electronic certificates meet standards specifying its content in a rigorous way. Electronic certificates can be used in various applications within the security of information systems to ensure: A non-repudiation and data integrity with digital signature or electronic signature (forward), and data privacy through encryption of data, authentication of an individual or a non-physical identity (Web Server - SSL Workstation - 802.1x, IPSec VPN - SSH - SSL, mobile code, electronic documents).

The certificates are widely used on E commerce sites, web mail or other sensitive sites (banking, taxes, E voting etc...). Multiple levels of encryption exist, and several associated features make the understanding of complex licenses. Usually the electronic digital certificate consists of: A serial number used to uniquely identify the certificate, subject which is the person or entity identified, signature algorithm that is used to create the signature, issuer which is the entity that verified the information and issued the certificate, valid-from that is the date the certificate is first valid from, valid-to which is the expiration date, key-usage that is the purpose of the public key (e.g. encipherment, signature, certificate signing...), public key, and thumbprint algorithm that used to hash the certificate and thumbprint which is the hash itself to ensure that the certificate has not been tampered with.

2.6 Related Works

This section includes most related papers with a brief summary about each:

(Wei-Chi. &Sheng-De, 1999): A secure E voting scheme is proposed by Wei-Chi.& Sheng-De under to overcome the problem of voting disruption if some voters abstain in the intermediate stage. Some assumptions that derive from the realistic environments (a) An anonymous channel exists, (b) A one-way permutation function exists, (c) RSA is secure, and (d) At least one scrutineer is responsible at any moment in the voting.

(Subariah et. al, 2001): A general E voting system is proposed that employs a cryptographic technique to overcome the security issues in the election process. The voter's privacy is guaranteed by using a blind signature for confidentiality and voter's digital signature for voter's authentication.

(Joaquim, Zúquete&Ferreira, 2002): A robust E voting system designed for distributed and faulty environments, namely the Internet. The goal of REVS is to be an E voting system that accomplishes the desired characteristics of traditional voting systems, such as accuracy, democracy, privacy and verifiability.

(Schryen, 2004): Schryen try to fix cryptographic procedures for ballots rather than proposes a new voting protocol by presenting a structural security framework for E voting. The fixing identifies the responsibilities and rights for the authorities involved or security precautions regarding hardware and software.

(Cetinkaya O. & Doganaksoy, 2007): The work by Cetinkaya & Doganaksoy aims at bringing unlinkable pseudo-voter identities based on blind signature bear on anonymous E voting protocols by presenting a Pseudo-Voter Identity (PVID) scheme based on blind signature to achieve anonymity in E voting protocols. Blind signature is applied on pseudo identities selected by voter. Therefore voter obtains blindly signed pseudo identities namely PVIDs and uses them throughout the entire communication with the authorities. By using PVID scheme, E voting protocols do not need anonymous channels anymore.

(Naznin, Dey, Bhuiyan & Saidur , 2007) : An efficient implementation for E voting is proposed that contains the automation of an online voting system providing some features which were absent in the previous implementations. The proposed implementation is more user friendly and secured but faster than the others using recent technologies and resources.

Meng, Li & Qin (2010): Meng, Qin develops a receipt-free coercion-resistant remote internet voting protocol based on MW deniable encryption scheme and bit commitment scheme. They include an analysis of receipts freeness and coercion resistance of the proposed remote internet voting protocol . Finally, they compare security properties of several typical protocols with proposes protocol.

(Lee, 2010): Lee presents an analysis of the procedure of an elementary E voting system using RFID technology, and discusses its security issues. A significant security issue that lee brings by relying on the RFID technology is making the recounting easy by separated the ballots from the voting software and hardware.

Chen, et al. (2011): Chen and others utilize the double-trapdoor commitment scheme to propose a new receipt-free voting scheme based on blind signatures for large scale elections. Also, the scheme presents a more efficient zero-knowledge proof for secret permutation. Therefore, the proposed scheme is much more efficient than Okamoto's schemes with the weaker physical assumptions.

(Kalaichelvi. & Chandrasekaran, 2011): Kalaichelvi. & Chandrasekaran propose a secure E voting protocol. Their suggested scheme does not require a special voting channel and communication can occur entirely over the current internet. This method integrates internet convenience and cryptology. Thus, the proposed scheme satisfies the more important requirements of any E voting scheme: completeness, correctness, privacy, security and uniqueness.

Chapter Three

The proposed E voting Scheme

This chapter proposes a new scheme that acts as an improvement over the last two, Evox-MA and REVS, E voting protocols that based on the blind signature. The researcher proposes an E voting scheme that is suited for large scale election, overcome above the problems associated in these protocols and achieve all E voting security requirements.

Like any E voting scheme, the proposed scheme consists mainly of three stages: The preparation stage, voting stage and counting stage. At each of these stages more than one cryptographic scheme or protocol or modified one is applied to provide some security measures. For example, in the preparation stage the modified PVID scheme is used. The benefit gains from applying the PVID is that the voter uses pseudo identities, which have no relation with the voter's real identity and are unlinkable to it, so voter can use them throughout the entire communication and he/she can easily hide his/her real identity and for certificate validation a responder entity is added to cooperate with a authentication server (AS) Kerberos authentication protocol component to assure that only authorized voters were voted during the specified election period. More and more cryptographic techniques is added to the proposed scheme, and consequently provided the required secure E voting scheme. Figure 3.1 present a conceptual point of view for the proposed scheme, each of these steps were explained in details at each stage later.

The proposed E voting scheme will depend on the public key encryption unless otherwise stated. Briefly at step 1, the commissioner will send a message encrypted with commissioner private key consist of the voting public key that used in overall voting process. At step 2, the commissioner will send a hash value for the voting public key using SHA-1 algorithm. In the preparation stage (step 3– step 10), the modified PVID scheme will be operated. In step 3, the voter will send a set of blinded identities M_b , after the ID generation and blinding PVID stages applied, to the PVID authority, it will never sign a non eligible voter as it will check the voter RegID against country election law. As the voter is eligible the PVID authority will sign a set of the voter blinded identities (M_{bs}) via a PVID signing stage and send them to the

voter accompanied with the issued voter certificate in step 4. The optional step for the voter to contact a password generator (PG) to generate a unique password for each voter, instead of using the traditional voter password, as an attacker may keep track of the voters' password and compromise it. From (step 5- step 10), the modified Kerberos authentication protocol will be operated with the converted Ferguson E cash protocol. In step 5, the voter will send a message encrypted with the AS public key consist of the voter certificate and a set of the signed blinded identities (PVID-list). As the AS receives this message at step 5, it will send to the responder to check its status in step 6, the OSCP-KIS will be applied to operate in a distributed environment. The responder will contact a PVID authority to check a certificate status in step 7; the PVID authority will send the voter certificate status to the AS in step 8, to the voter via AS in step 9, 10. A Kerberos authentication protocol consists of other steps that eventually end with the generated voter authenticate ticket that will be used in the voting stage, administrators will never sign a voter without the Kerberos authenticated ticket.

In the voting stage (step 11- step 17), the converted Ferguson E cash protocol continue to operate. In step 11- step 14, the voter will get the ballot that consists of the candidates from which the voter can choose his/her vote. A bulletin board will be applied so the voter can check the received ballot (step 14) by calculate the same hash for the received ballot and compare it with the one associated with EBG bulletin board. In step 16, the voter will contact a dealer and send the Kerberos authenticated ticket with the PVID-list, as a dealer verify these received component, it will send to the voter the administrators public key, so the voter can encrypt his/her vote. Noted that a PVSS will be applied to avoid the collude administrators problem as will be explained later.

In the counting stage, the voter will commit to the ballot in the message commitment trap door commitment phase and send the committed ballot anonymously to the counter for counting purpose (step 18); more details will be provided in the counting stage later.

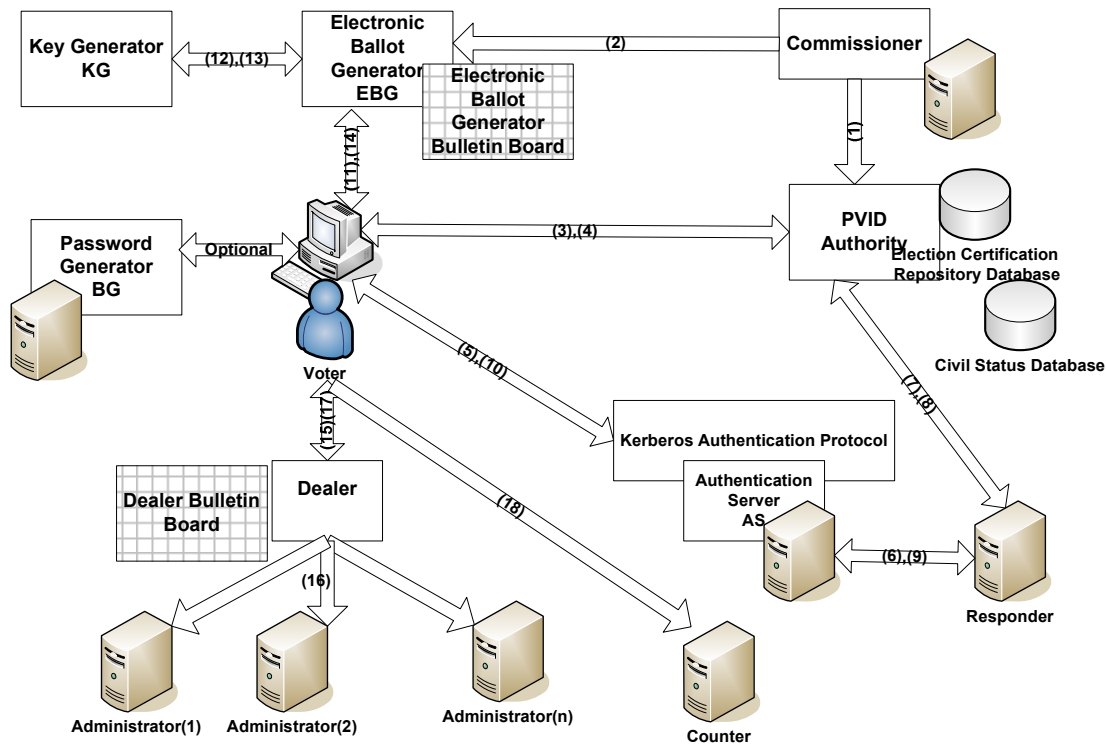


Figure 3.1: A conceptual point of view for the proposed

At first, the researcher will present in details each E voting stage in terms of entities involved and cryptographic technique applied and the benefits gain of such application or modification. The notation used in the proposed scheme will be described where it is referenced.

3.1 Preparation Stage:

In order to achieve voter privacy at E voting protocol, the researcher applied here PVID scheme. In PVID scheme, voter prepares a list of blinded identities and then he/she obtains blind signature for each of them separately by interacting with the approval authority in one session. Later, voter extracts anonymous pseudo identities (PVIDs) which are unlinkable to voter registration identity. Each of PVID is selected by the voter and blindly signed by the approval authority after verifying voter eligibility. The value of PVID is only known by the voter.

In existing voting protocols (Chaum 1981, 1983, Fujioka, Okamoto & Ohta, 1992, Cohen & Fischer, 1985; Benaloh 1987), voter generally uses his/her real identity while communicating with the authorities. While, in PVID scheme, voter uses pseudo identities, which have no relation with the voter's real identity and are

unlinkable to it. Voter can use them throughout the entire communication and he can easily hide his/her real identity. Some modification had been applied to the used PVID scheme such as the voter issue certificate to provide more secure scheme, as will be discussed later.

Later, by using the modified Kerberos authentication protocol, the Authentication server will verify the eligible voter issued certificate by using the OSCP protocol (as the voting is online the CRL can't be used, so the OSCP will be operated under the distributed environment here, with a hash function applied additionally for a timeliness checking purpose .

3.1.1 PVID Scheme

Voter has a registration identity (RegID) which can be any widely used identity such as national identity number or social security number. RegID can be a government-issued voter ID as well. On the Election Day, voter uses his/her RegID to authenticate himself to the system. In almost all blind signature based voting protocols, voter tries to obtain blindly signed ballot and/or his/her cast or part of them. In PVID scheme, voter obtains a list of blindly signed anonymous pseudo identities and uses them instead of real RegID while interacting with the authorities.

The PVID, the responsible authority, issues a blind signature on voters PVID-list after checking voter eligibility. The trust of this authority is very important as it can blindly sign ineligible voters PVID list. As soon as the voter obtains a PVID-list, he/she can use in later communication instead of using the voter RegID (public key) as this will be vulnerable for attack. By applying the PVID scheme in the proposed scheme, the privacy degree will be increased. Whatever, PVID is consider as one of the most practical scheme as it apply only blind signature to obtain the authority signature.

It provides as well privacy without requiring any complex mechanisms and computational operation. RSA is used as a public key cryptosystem. A pseudo random number generator is used to feed PVID with random number. By using the *elliptic key cryptography*, the voter will generate his/her associated key pairs, public and private keys (d_v, e_v) as the following:

$E_q(a,b)$: elliptic curve with parameter a,b and q ,where q is a prime or an integer of the form 2^m

G point on elliptic curve whose order is large value n

Voter pair key generation	
Select private d_v	$d_v < n$
Calculate public e_v	$e_v = d_v \times G$

By using cryptographically DES cyclic encryption random number generation which generates random numbers (see Figure 3.2). A counter with period N provides input to the encryption logic. For example, if a 56-bit DES keys are to be produced, then a counter with period 2^{56} can be used. After each key is produced the counter is incremented by one .Thus, the pseudorandom numbers produced by this scheme cycle through a full period: Each of the output X_0, X_1, \dots, X_{N-1} is based on a different counter value and therefore $X_0 \neq X_1 \neq \dots \neq X_{N-1}$. Because the master key is protected, it's not computationally feasible to deduce any session keys (random numbers) through knowledge of one or earlier session keys.

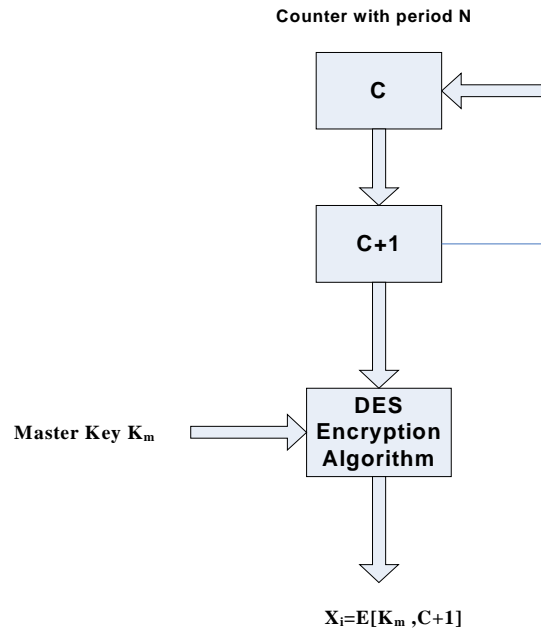


Figure 3.2: Pseudorandom number generation from a counter

PVID scheme has four stages: ID generation stage, blinding stage, signing stage and PVID obtaining stage. The detailed descriptions of these stages will be described as the following:

- **ID Generation Stage**

Voter generates k pseudo identity numbers and prepares ID-list. Each ID contains the election data, authority data and a big random number (generated by a PRNG shown in Figure 3.2, so it is constructed as follows; for each ID, the authority data should be different whereas the random number should be same. Using same random number provides that IDs belong to one voter.

$$ID_i = (\text{Election Data}, \text{Authority Data}, \text{Random Number})$$

$$\text{ID-list} = \{ID_1, ID_2, \dots, ID_k \mid ID_i \text{ is } i^{\text{th}} \text{ pseudo identity}\}.$$

Now, voter has an ID-list that he wishes to have signed each ID_i in the list by PVID Authority. Voter does not want PVID Authority to learn anything about ID_i . More details are indicated about ID-List in Figure 3.3.

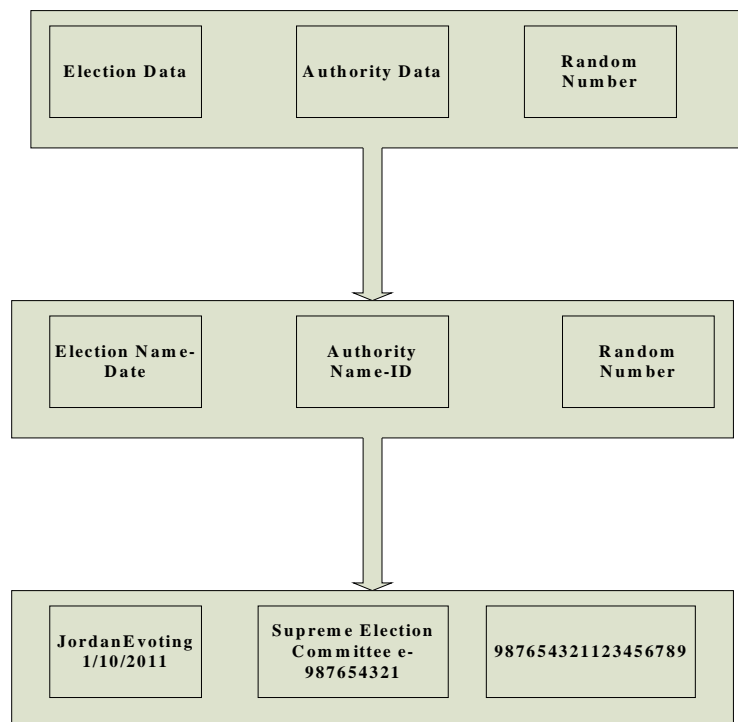


Figure 3.3: ID-List Details (Cetinkaya O. & Doganaksoy, 2007)

- **Blinding Stage**

Voter generates a random blinding factor number r (using PRNG shown in Figure 3.2) and calculates blinded message m_b for each ID_i , and obtains a list of blinded IDs which is M_b as shown in Figure 3.4.

$$m_{bi} = (r^e [ID_i] \bmod n) \quad \text{Where } \gcd(n, r) = 1 \dots\dots\dots (3.1)$$

$$M_b = \{m_{b1}, m_{b2}, m_{b3}, \dots\dots\dots m_{bk}\} \dots\dots\dots (3.2)$$

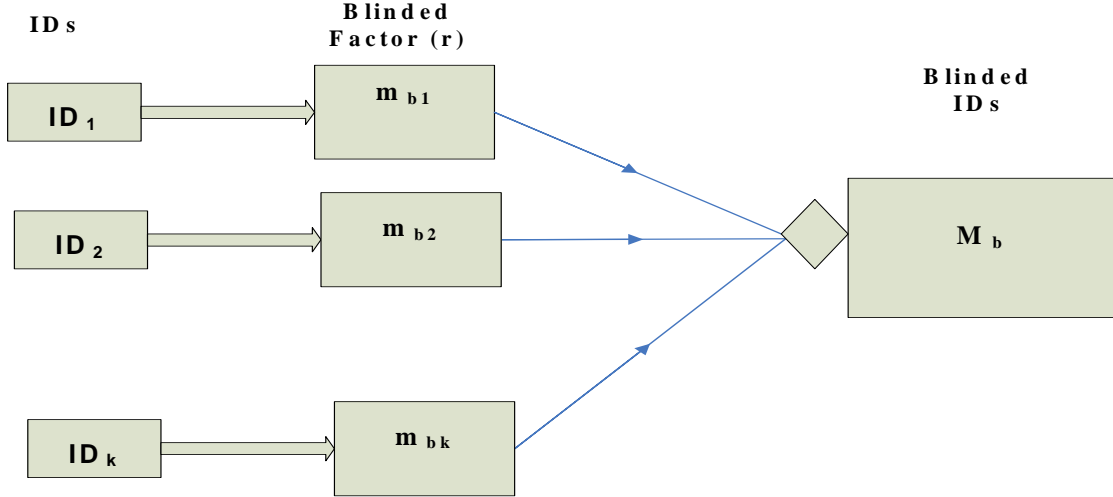


Figure 3.4: Blinding stage (Cetinkaya O. & Doganaksoy, 2007)

Voter signs the list M_b and obtains $d_v(M_b)$. Then, he/she will encrypt his/her RegID and $d_v(M_b)$ with PVID authority public key ($PU_{PVID-Authority}$) and obtain $E(PU_{PVID-Authority}(RegID, d_v(M_b)))$. Voter will send this message to PVID authority (see Figure 3.5). As the value m_b is blinded by the random value r , it can't derive any useful information from it. This message will accompany with another message that contains the following $\{e_v, E(d_v((RegID)_v))\}$. Noted that the voter send his/her RegID to let the PVID authority check the RegID against country election registration laws. The voters' public key (e_v) is sending in clear for two later purposes. First, for checking voter signature. Second, for encrypting the list of blinded voter identities if he/she permitted to vote.

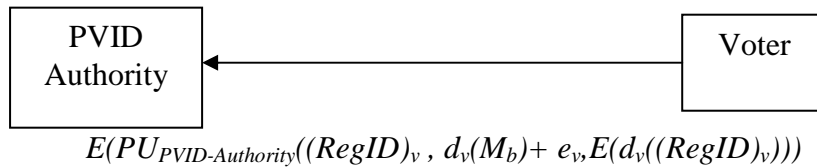


Figure 3.5: Voter-PVID authority interaction

- **Signing Stage:**

PVID authority will decrypt the received message, and obtains the voters *RegID* and $d_v(M_b)$. It will verify the voters' eligibility by checking his/her *RegID* against the civil status data base. If the voter is eligible and hasn't made any request yet, PVID uses voter public key (e_v) and check the voter signature on M_b . For each eligible voter, PVID authority signs each blinded message m_b in the list of M_b and calculates m_{bs} . Subsequently; PVID authority obtained a list of blindly signed IDs which is M_{bs} . As indicated in Figure 3.6.

$$m_{bs_i} = m_{b_i}^d \bmod n \dots \dots \dots (3.3)$$

$$M_{bs} = \{m_{bs_1}, m_{bs_2}, \dots, m_{bs_k}\} \dots \dots \dots (3.4)$$

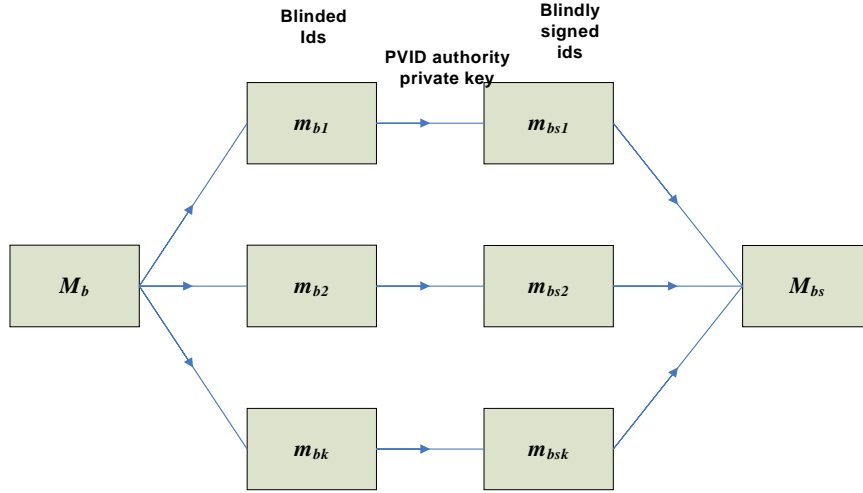


Figure 3.6: Signing Stage (Cetinkaya O. & Doganaksoy ,2007)

Then, PVID authority encrypt the list M_{bs} with the voter public key ($e_v(M_{bs})$), for PVID authority to supply only one PVID for each eligible voter it will change the voter status and issue a voter certificate (review the literature for public key certificate), $Cert_v = E(PR_{PVID-Authority}(e_v(\text{Time}_1 \parallel \text{RegID} \parallel \text{ElectionData} \parallel e_v)))$ these will be send to the voter (Figure 3.7 (step(1))), also this will be accompanied with PK_{voting} , that the PVID authority received from the commissioner (Figure 3.7 (step(2))). The PK_{voting} , the key used in overall voting and know to each involved entity, was encrypt with the commissioner private key $E(PR_{Commissioner}(PK_{voting}))$, as it received the PVID authority will decrypt it with the commissioner public key

($PU_{\text{commissioner}}$), confidentiality and authentication between communicating entities will also be achieved here by such an encryption and decryption operations. Furthermore; the commissioner will send a hash for voting public key (PK_{voting}) for verification purpose to the Electronic Ballot Generator (EBG), as shown in Figure 3.7 step (3)). Noted that all hash values will be introduced using SHA-1 cryptographic algorithm.

As the voter status had been changed and the certificate was issued to the eligible voter, this will achieve the E voting requirements of democracy and completeness, a copy of the certificate will keep in the repository. So if a voter try to vote again, any such attempt will be easily detected either by checking voter status (vote or unvote) in civil status database or by checking the issued voter certificate in election certification repository database, depend on that the certificate obtains only once (the issued certificates are kept in database) for authorized voters only, which permit voter to participate in election during the specified election period, and send back a component [A] $PR\text{-}PVID_{\text{Authority}}$, so the voting process is canceled. Otherwise (in case the voter hadn't voted before) the value component [O] $PR\text{-}PVID_{\text{Authority}}$ is sent and voting registration continued.

Another way is provided here in order to detect a double voting attempt, the PVID authority supply only one PVID for each eligible voter and doesn't make any sign for the blinded identities if the voter had been signed before. The issued voter certificate will be multi-encrypted with voter public key (e_v), public key encryption (asymmetric encryption) will be used here, and PVID authority private key ($PR_{PVID\text{-}Authority}$), as shown in Figure 3.7.

$$(1)(e_v(M_{bs}, PK_{voting}) + Cert_v = E(PR_{PVID-Authority}(e_v(Time_1 \parallel RegID \parallel ElectionData \parallel e_v)))) + [O]_{PR-PVIDAuthority}$$

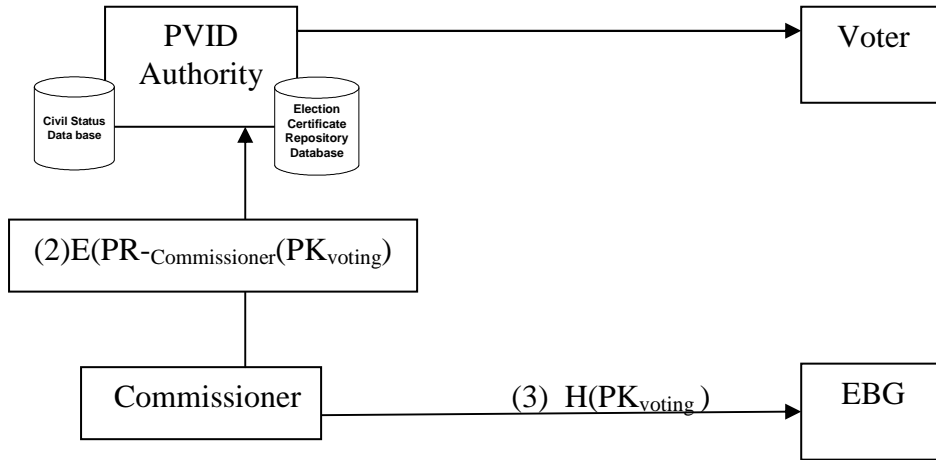


Figure 3.7: Message passing in signing stage

- **PVID Obtaining Stage**

As the voter received the blindly signed ID list M_{bs} . He/she will decrypt them and can easily now obtain PVIDs, the true sign of IDs, by removing the blinding factor r from each m_{bs} . Voter carries out the following operations for each m_{bs} in the list M_{bs} in order to obtain $PVID_i$ for each ID_i . Also the voter will obtain his/her certificate now, by decrypting it using the PVID Authority public key ($PU_{PVID-Authority}$) and make sure about the decrypting information ($Time_1 \parallel RegID \parallel ElectionData \parallel e_v$) by this he can trust such authority.

$$m_{bs_i} = m_{b_i}^d \mod n = (r^e [ID_i])^d \mod n \dots \dots \dots (3.5)$$

$$m_{bs_i} = r^{ed} [ID_i]^d \mod n = r [ID_i]^d \mod n \dots \dots \dots (3.6)$$

$$PVID_i = r^{-1} m_{bs_i} \mod n = [ID_i]^d \mod n \dots \dots \dots (3.7)$$

$PVID_i$ is the sign of PVID authority on the voters selected ID_i . Then, the voter will calculate PVID-list with PVID as the Figure 3.8 shows.

$PVID-list = \{ PVID_1, PVID_2, \dots, PVID_k \}$

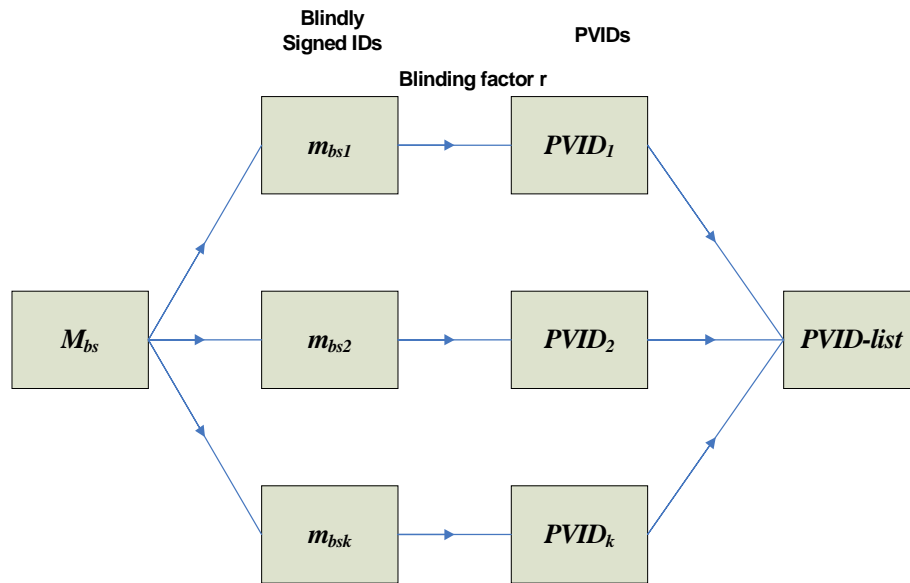


Figure3.8: PVID obtaining stage (Cetinkaya O. & Doganaksoy ,2007).

Now, voter has valid and signed pseudo identities that are unlinkable to his/her real RegID. Voter can use them in the proposed E voting scheme without providing his/her RegID to the voting authorities. Moreover, he/she can directly communicate with the authorities without requiring any anonymous channel since PVIDs aren't linkable to his/her real identities.

When voter uses his/her PVID, the authority only verifies the signature on PVID by unsigning it with PVID authority public key and simply checking the election data and authority data. Noted that the same strategy had used under E cash environment to assure a non repudiation service as (spognardi 2006) indicated in his survey. Here; it had been used according to PVID scheme (Cetinkaya& Doganaksoy ,2007) in E voting environment. As explained above some modification had been applied (e.g. the voter issue certificate) and other more in order to provide the secure E voting scheme.

The optional step associated with the proposed scheme that preferred by the voter to contact a password generator (PG) that is responsible to generate a unique password for each eligible voter, instead of using his /her own traditional password that usually he/she used in other website (attacker may keep track of a user password and compromise the voter password). The voter will send $\{e_v, PVID-list\}$ to password generator (PG) as Figure 3.9 show.

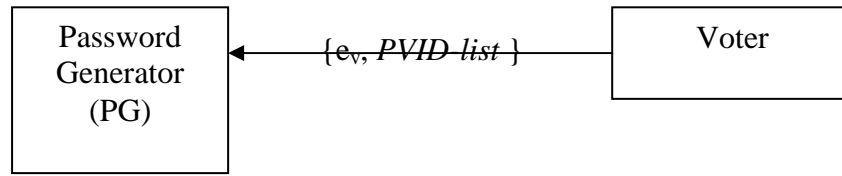


Figure 3.9: Voter-PG interaction

The password generator is responsible to generate a unique password for each eligible voter, the following algorithm (Kelsey et al., 1997) can be used to generate such passwords under E voting environment, as well as the PG received the signed PVID authority pseudo identity (*PVID-list*), it will verify the PVID authority sign and signed it again with a PG private key (PR_{PG}), for non-repudiation goal [*PVID-list*] $_{PR-PG}$, so the voter can trust a such generator.

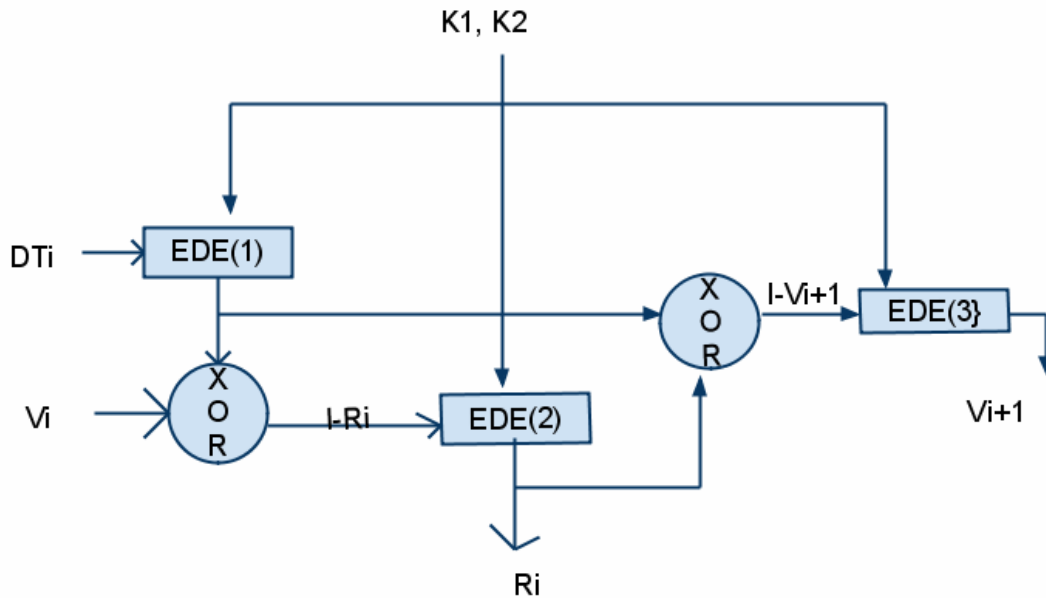


Figure3.10: ANSI X9.17 for pseudorandom number generator(Kelsey et al., 1997)

Table 3.1: Password generator algorithm Description

DT_i	Datre /time at the beginning of ith generation stage
$(V)_i$	Combination of voteridentity and public key at round i
R_i	Finally generated password
K_1, K_2	DES key used at each round
Then : $R_i = \text{EDE}([k_1, k_2], [V_i \text{ [XOR] EDE}([K_1, K_2], DT_i)])$ $V_{i+1} = \text{EDE}([k_1, k_2], [R_i \text{ [XOR] EDE}([k_1, k_2], DT_i)])$ Where $\text{EDE}([k_1, k_2], X)$ refers to the sequence encrypt-decrypt-encrypt using two key triple DES to encrypt X .	

The password generator algorithm will depend on the ANSI X9.17 PRNG cyclic generated random number encryption infrastructure (see Figure 3.10, Table 3.1). It will use triple DES for encryption . The ingredients are as follows :

- **Input :**
 - ✓ 64 bit representation of current date and time, which is updated at each generation.
 - ✓ 64 bit representation which is a combination of voter public key and signed blinded identities ($e_v + \text{PVID-list}$) that differs at each round (each vote has a different identity).
- **Keys :**

Making use of the three triple DES encryption modules, with a 56 bit keys, which must be kept secret and are used for password generation.

- **Output :**
The output consist of a 64-bit for password (R_i) and a 64-bit seed value .

Several factors contribute to the cryptographic strength of the proposed approach. It involve a 112-bit key and three Encryption Decryption Encryption (EDE) for a total of nine DES encryption. It is driven by two input, the date and time values and the voters public key with the PVID-list, which will be differ at each round. So, the amount of material that must be compromised by an opponent is overwhelming

even if R_i is compromised it would be impossible to deduce V_{i+1} from the R_i because an additional EDE operation is used to produce the V_{i+1} .

After the password is generated, it will be encrypted with the voter public key, and sent to the voter, this will also be accompanied with the password generator (PG) sign for the signed PVID authority ($[PVID-list]_{PR-PG}$), as shown in Figure 3.11.

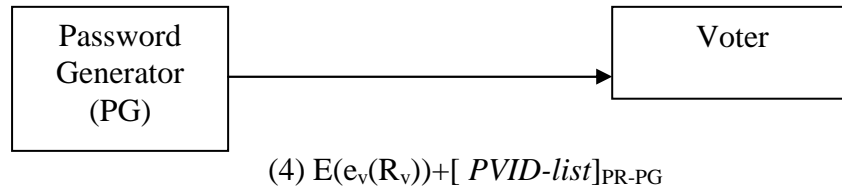


Figure 3.11: PG –Voter interaction

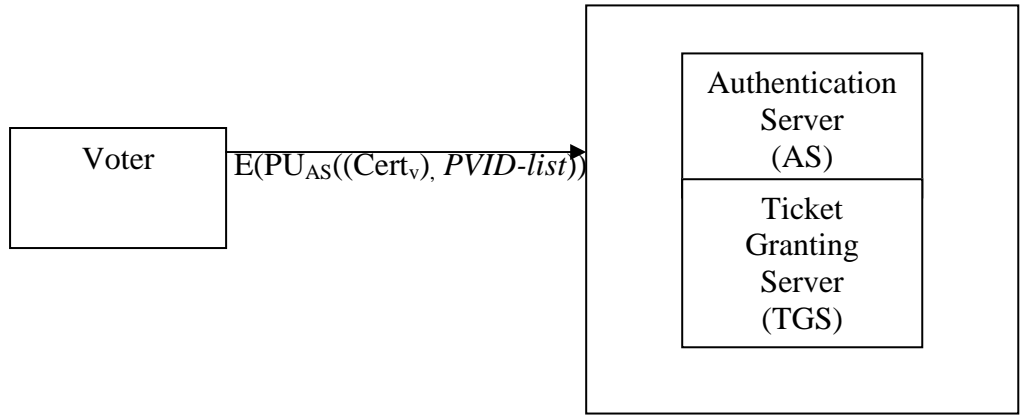
As the voter receives the generated password, encrypted with the voter generated public key (e_v), it will be decrypted using voter private key $D(d_v(R_v))$ and get the generated password. Also the voter will verify the PG sign $[PVID-list]_{PR-PG}$ using the password generator public key (PU-PG). By this way the confidentiality and a trust between communicating entities is achieved. Till now the voter securely has the generated $Cert_v$, e_v , d_v , and the signed blinded voters identities and R_v .

3.1.2 Kerberos authentication protocol under E-voting?

By applying Kerberos authentication protocol (review the literature) in the E voting proposed scheme, the researcher will guarantee that only authorized voter will vote, this will help in filtering the counter buffer from unauthorized votes and thus limit the DoS attack against attackers whom try to fill the counter buffer with garbage votes. In addition to the main Kerberos authentication protocol entities, some other entities will be added to it, in other words some modification will be applied for enhancing the security measures in the proposed scheme, such as a responder which derived from the D-OSCP protocol, the AS will interact with the responder which responsible to verify the validity of the certificate ($Cert_v$). For that any attempt from voter to supply the AS with expired or fake or old certificate will be easily detected here. Also any other attempt from voter to provide fake signed identities will be detected as it will be decrypted with the PVID authority private key. This will make the proposed scheme more secure than others.

By relying on Kerberos in the proposed scheme the researcher add a strong wall of protection and confidentiality by let the voters at first authenticated by the PVID authority issued certificate in the preparation stage ($Cert_v$), rather than the voter password, and then let the voter and any other Kerberos communicating entities to share a secret key based on the **Nonce Based Authentication Scheme**, instead of the shared secret key based on the voter RegID and password (R_v). Accompanied with the Kerberos authentication protocol, the Ferguson E cash protocol, authentication part, had been modified to operate under E voting, this will add more sophisticated authentication measures to the proposed scheme.

In the proposed E voting scheme, the researcher rely on Kerberos version 5 that specified in RFC 1510 (review the literature). Here, the researcher will present how the Kerberos will be operated under E voting environment taking the modification into consideration. The researcher presents a step by step interaction (message exchanged) between a voter and a Kerberos communicating entity. First, the voter will send a message which consists of his/her own generated certificate ($Cert_v$) and identities signed with the PVID authority to the AS encrypted with the AS public key (PU_{AS}) (see Figure 3.12).



Kerberos Protocol

Figure 3.12: Voter-AS interaction

As the AS receives this message, it will decrypt it using its own private key and obtain the associated encrypted information (*PVID-list*), $Cert_v$. Then, AS will verify the signed identities. Later, the AS will verify the $Cert_v$ by contact a responder. The AS will send a request containing $Cert_v$, encrypted with responder public key (PU_{res}), to the responder for $Cert_v$ verification purpose. As the responder receive the message, it will be decrypted using the responder private key (PR_{res}) (see Figure 3.13). Then, the responder will contact the PVID authority that issue such a certificate for verification purpose.

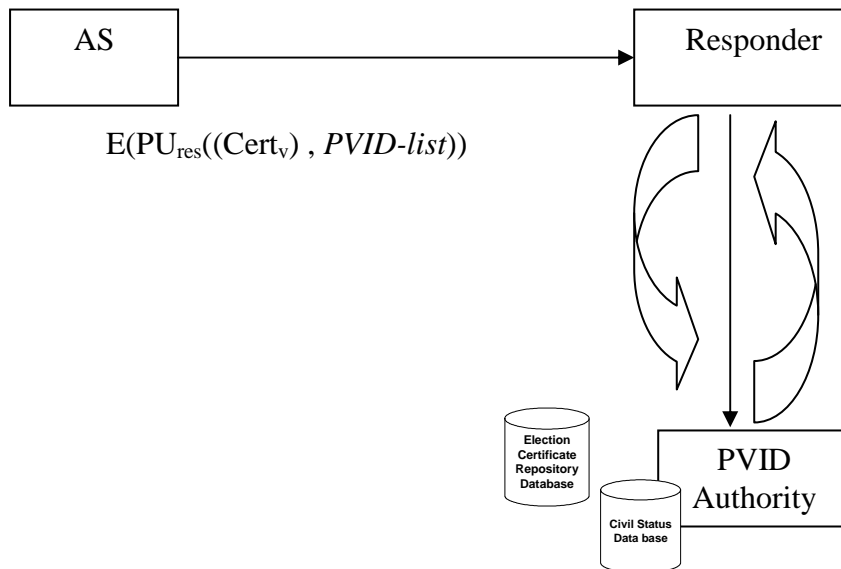


Figure 3.13: AS-Responder interaction

Either if the certificate is valid or revoked the PVID authority will send a response to the responder contain the status of the certificate. In order for the responders to operate in a distributed environment, the researcher adopts the scheme based on *KIS-OSCP* and applying a hash additionally (Lagnana A. et.al (2004)) to let a responder server operate in a distributed environment. By using different private keys but just one corresponding public key is possible. So only one certificate is needed for every responder and to check the timeliness of responder, hash chain is used. Let simply explaining how the OSCP-KIS key insulated signature scheme will be operated in the proposed scheme. Typically the size of t is similar to that of n . assuming that $(n-1, n)$ key insulated signature scheme is used. Let R_1, R_2, \dots, R_n be the n designed responders. When an AS receives a response from R_i , the AS should verify the response as follows:

- (1) The AS checks the revocation status of the responder R_i 's public key.
- (2) The AS verifies the digital signature contained in the response by using the responder R_i 's public key.

The step (2) consists of two main stages:

- **Key generation:**

To generate and distribute every responder's private key for digital signature, PVID authority chooses a master secret and calculates its corresponding public key. Then, if the number of responders is n , PVID authority generates n private keys for responders by applying KIS key generating algorithm and securely distributes the keys to each responder. In the key generation, the PVID authority will distribute private keys for every responder as Figure 3.14 shows.

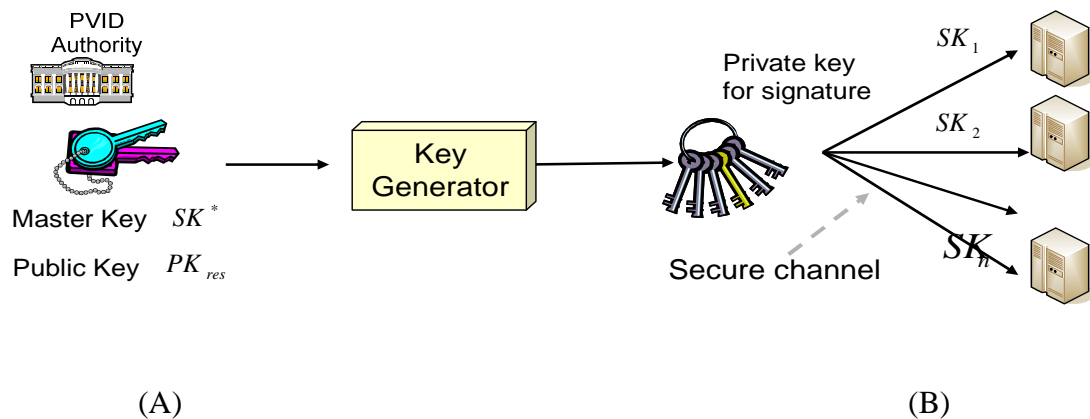


Figure 3.14: KIS-OSCP key Generation

Theoretically, in (A):

For public key generation :let p and q prime numbers such that $p=2q+1$ and g ,h be the elements of order q in Z_p .The PVID authority will generate a master key $SK^*=(x_0^*,y_0^*,\dots,x_{n-1}^*,y_{n-1}^*)$ by choosing $x_i^*,y_i^* \in S_o$ randomly . SK^* is used for private key generation .Responders' public key $PK_{res}=(g,h,v_0^*,\dots,v_{n-1}^*)$ is calculated by $v_i^*=g^{x_i^*}h^{y_i^*}$.

$$(x_0^*,y_0^*,\dots,x_{n-1}^*,y_{n-1}^*) \leftarrow Z_q$$

$$v_i^*=g^{x_i^*}h^{y_i^*} \text{ mod } p \quad \text{for } 0 \leq i \leq n-1 \dots\dots\dots(3.8)$$

$$SK^*=((x_0^*,y_0^*,\dots,x_{n-1}^*,y_{n-1}^*)) \dots\dots\dots(3.9)$$

$$PK_{res}=(g, h, v_0^*,\dots,v_{n-1}^*) \dots\dots\dots(3.10)$$

In (B): A private key will be generated :a different private key is assigned to each responder with the initial value of $SK_0=(x_0,y_0)=(x_0^*,y_0^*)$,the responder Ro's private key Sk_i is generated as follows :

$$x_i' = \sum_{k=1}^{n-1} x_k^* (i^k - (i-1)^k) \dots\dots\dots(3.11)$$

$$y_i' = \sum_{k=1}^{n-1} y_k^* (i^k - (i-1)^k) \dots\dots\dots(3.12)$$

$$x_i = x_{i-1} + x_i' \dots\dots\dots(3.13)$$

$$y_i = y_{i-1} + y_i' \dots\dots\dots(3.14)$$

$$SK_i = (x_i, y_i) \dots\dots\dots(3.15)$$

- **Hash chain**

The PVID authority will deliver the private key Sk_i to R_i anonymously .After all private keys are derived, intermediate values including the master key $SK^*=(x_0^*,y_0^*,\dots,x_{n-1}^*,y_{n-1}^*)$ are deleted .

Then, PVID authority generates hash chains to be used for timeliness checking. If the total time periods are T, PVID authority generates T chained hash values for each responder and keeps the first elements securely. Each hash value is used for given time period. If the time period is one day, 365 hash values are

generated per responder. AS checks the timeliness of a responder by checking (hash chain) at the given time period.

PVID authority issues the certificate for all responders. This certificate includes KIS public key and the first hash values in the hash chain of all responders.

$$X_1 = H(X_2) = H^2(X_3) = \dots H^{t-1}(X_t)$$

For total T time period and n responders :

$$\left\{ \begin{array}{l} X_T^1 \rightarrow X_{T-1}^1 \rightarrow \dots X_t^1 \rightarrow \dots X_1^1 \\ X_T^2 \rightarrow X_{T-1}^2 \rightarrow \dots X_t^2 \rightarrow \dots X_1^2 \\ \dots \\ X_T^n \rightarrow X_{T-1}^n \rightarrow \dots X_t^n \rightarrow \dots X_1^n \end{array} \right.$$

PVID authority keeps them securely .PVID authority provides X_t^i at time period $t \in T$ to i-th responder ,the validity checks at $t \in T$ for i-th responder ,the value to be checked ($X_1^i = H^{t-1}(X_t^i)$) is true (in signing and verification phase).

- **Signing /Verification Algorithm:**

(1) Signing Algorithm :When R_i sends a response to AS , R_i generates a digital signature (i,w,a,b) by using $SK_i=(x_i,y_i)$ as follows :

$$r_1 r_2 \leftarrow Z_q$$

$$w = g^{r_1} h^{r_2} \bmod p \dots \dots \dots (3.16)$$

$$r = H(i, M, w) \dots \dots \dots (3.17)$$

$$a = r_1 - \alpha x_i \dots \dots \dots (3.18)$$

$$b = r_2 - \alpha y_i \dots \dots \dots (3.19)$$

Where $H(\cdot)$,is a cryptographic hash function

(2) Verification Algorithm : The AS will verify the R_i 's signature (i,w,a,b)by using $PK_{res}=(g,h,v_0^*, \dots v_{n-1}^*)$ as follows :

$$v_i = \prod_{k=0}^{n-1} (v_i^*)^{i^k} \bmod p \dots \dots \dots (3.20)$$

$$\tau = H(i, M, w) \dots \dots \dots (3.21)$$

$$w = g^a h^b v_i^\tau \bmod p \dots \dots \dots (3.22)$$

As shown by Figure 3.15.

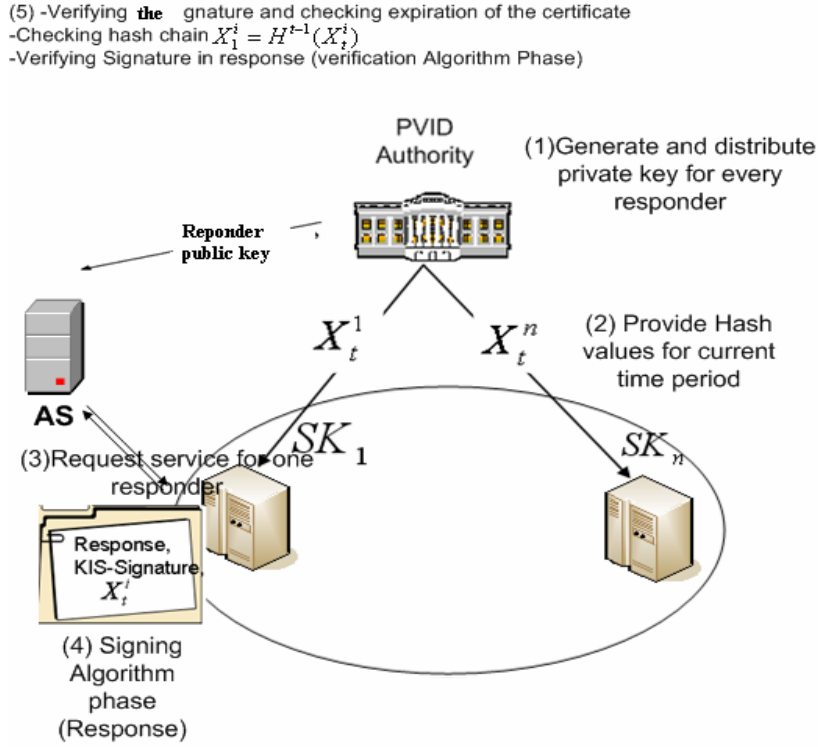
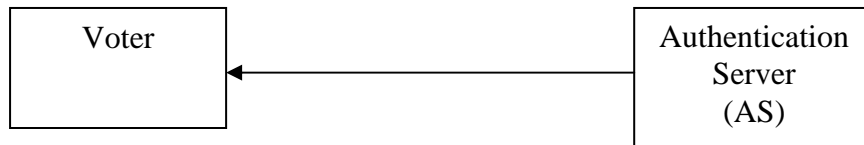


Figure 3.15: D-KIS-OSCP phases

By this the AS verifies the $Cert_v$ validity, then it will send the voter certificate again to the voter, after keep a copy of it in its own database, this is another way for detecting any attempt for double voting by a voter, with the update time and signed PVID authority identities signed again with AS private key, all entities of this message will be encrypted with AS private key $\rightarrow E(PR_{AS}(Cert_v, time + 1, (PVID-list)_{PR-AS}))$ (see Figure 3.16).



$$E(PR_{AS}(Cert_v, time + 1, (PVID-list)_{PR-AS}))$$

Figure 3.16: AS-Voter interaction 1

After the voter receives the previous message, it will be decrypted using PU_{AS} and the voter check the time from his/her own certificate and the received update one so he/she can judge any attempt of forgery, also he/she will verify the signed of *PVID-list*. Instead for AS to generate the shared secret key based on a combination of

voter's identity and password that both know. The voter and AS can authenticate each other and agree on a session key to be used between them based upon **Nonce Authentication scheme** (Yang et al.,2000) as follows:

(1) $V \rightarrow AS$: Request (ID_v, N_c)

The user generates a random number N_c and sends Request(ID_v, N_c). To generate such N_c a method shown in Figure 3.2, can also be used here.

(2) $AS \rightarrow V$: Challenge (realm, $N_s \text{ (XOR) } h(R_v \parallel N_c), h(R_v \parallel N_s \parallel N_c))$.

As the AS receives the Request message, the AS generates a random N_s and uses N_s, N_c, R_v to compute $N_s \text{ (XOR) } h(R_v \parallel N_c)$. Then, the server uses R_v, N_s, N_c to compute $h(R_v \parallel N_s \parallel N_c)$ and sends Challenge (realm, $N_s \text{ (XOR) } h(R_v \parallel N_c), h(R_v \parallel N_s \parallel N_c))$ to the voter.

(3) $V \rightarrow AS$: Response ($ID_v, \text{realm}, h(N_s \parallel R_v \parallel N_c)$)

When the voter receives the response message ,this voter uses N_c, R_v to compute $h(R_v \parallel N_c)$ and uses $h(R_v \parallel N_c), N_s \text{ (XOR) } h(R_v \parallel N_c)$ to compute $h(R_v \parallel N_c) \text{ (XOR) } N_s \text{ (XOR) } h(R_v \parallel N_c)$ to get N_s . Then, the voter uses R_v, N_s, N_c to compute $h(R_v \parallel N_s \parallel N_c)$.

If the computed $h(R_v \parallel N_s \parallel N_c)$ isn't the same as challenge ($h(R_v \parallel N_s \parallel N_c)$), the voter will be rejected by AS request $[A]_{PR-AS}$. Otherwise, the voter uses N_s, R_v and N_c to compute $h(N_s \parallel R_v \parallel N_c)$ and sends response ($ID_v, \text{realm}, h(N_s \parallel R_v \parallel N_c)$) to the AS server.

(4) When the AS receives response message, the server uses N_s, R_v, N_c to compute $h(N_s \parallel R_v \parallel N_c)$. If the computed $h(N_s \parallel R_v \parallel N_c)$ isn't the same as response ($h(N_s \parallel R_v \parallel N_c)$), the AS will reject such vote $[A]_{PR-AS}$. Otherwise the server accepts voter request $[O]_{PR-AS}$.

(5)After the AS and the remote voter authenticate each other, they use N_s as a session key between them SK_{V-AS} .

By this way both the AS and voter authenticated each other and agree on the session key to be used between them. The AS will send to the voter the following:

- Message A: The TGS session key that will be used between voter and TGS encrypted with the agree on voter and AS session key .
- Message B: Ticket granting Ticket (TGT) that consist of the *PVID-list* and voter network address (NW address_v), ticket validity period and voter TGS session key) encrypted with TGS secret key, which mean that the only one which can decrypt is the TGS (see Figure 3.17).

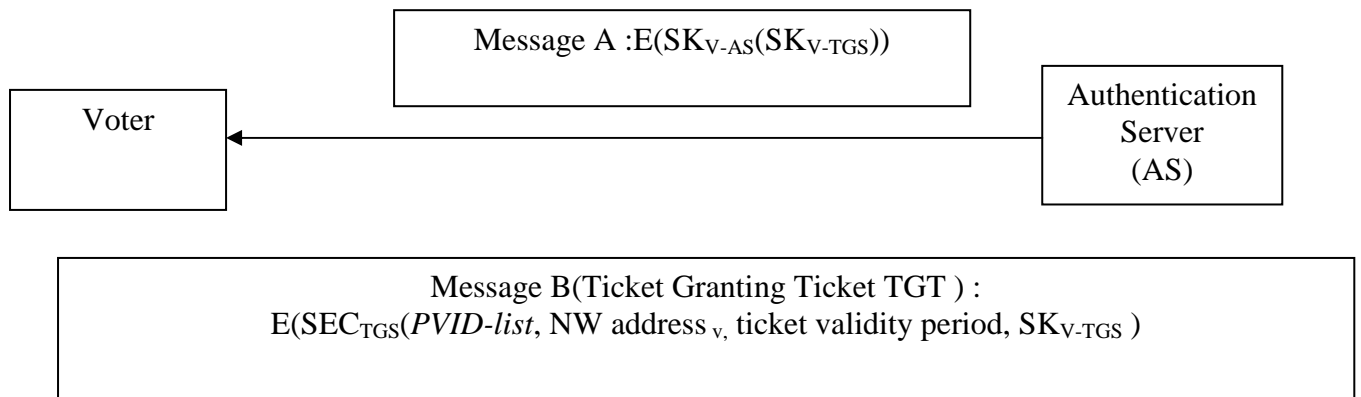


Figure 3.17: AS-Voter interaction 2

As the voter receives these two messages, he/she can deal only with message A, voter can decrypt it using the agree on voter and AS session key (SK_{V-AS}) as the following $D(SK_{V-AS}(\text{Message A}))$ and retrieve the associated session key that will be used between voter and TGS (SK_{V-TGS}) later. Noted that the voter can't decrypt message B as it encrypt with TGS secret key, which mean the only one that can decrypt is the TGS itself . Now, the voter will send two messages to TGS (see Figure 3.18):

- Message C: Same as the Message B, this received from AS by voter. With an election data (date in which election take place as state in PVID scheme this will be act as a voting service ID, known publicly to each eligible voter).
- Message D: An authenticator which consist of *PVID-list* and a timestamp (e.g. may indicated the current date and time) encrypt with the voter TGS session key retrieved from message A, in the step before.

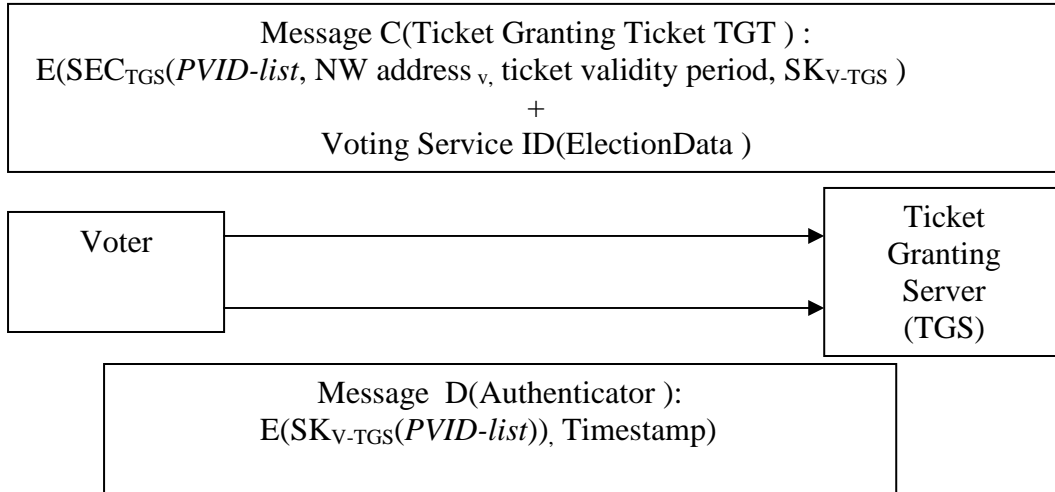


Figure 3.18: Voter-AS interaction

As the TGS received these two message (Message C and D), the TGS can decrypt message C using TGS secret key and therefore obtain the associated TGT, D ($\text{SEC}_{\text{TGS}}(\text{TGT})$). As it decrypted TGS can obtain the associated voter TGS session key. In this way both voter and TGS securely obtain the voter TGS session key and can talk with each other using $\text{SK}_{V\text{-TGS}}$.

Additionally, TGS will decrypt the authenticator (Message D) as it has the associated voter TGS session key, $\text{SK}_{V\text{-TGS}}$ from message C, that will be used in decryption operation $D(\text{SK}_{V\text{-TGS}}(\text{authenticator}))$ by TGS. After TGS decrypt these two message (Message C, D), it will make a match (if ($PVID\text{-list}$ from Message C == $PVID\text{-list}$ from Message D && $\text{Timestamp.Message D} \leq \text{Ticket Validity period.Message C}$)). Also it will verify the PVID authority signed pseudo identities. The TGS will now send two messages to the voter (see Figure 3.19)

- Message E : Is the voter to B-Voting server ticket that consists of ($PVID\text{-list}$, NW address $_v$, ticket validity period, $\text{SK}_{V\text{-B-VotingServer}}$) Encrypted with B-Voting server secret key ,the only entity that can decrypt is B-Voting Server itself
- Message F: is a voter B-Voting server session key encrypted with the voter TGS session key from A ($\text{SK}_{V\text{-TGS}}$)

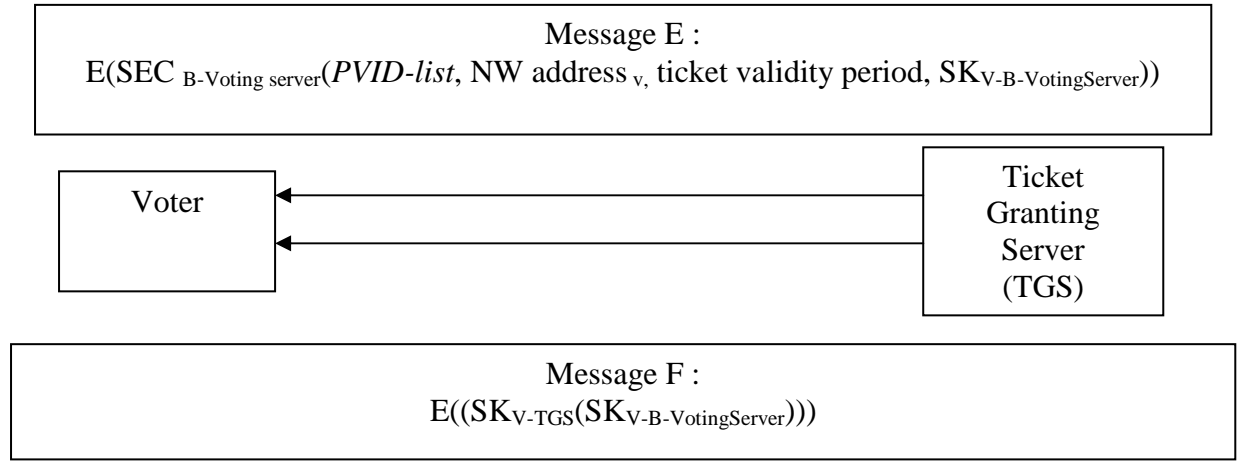


Figure 3.19: TGS-Voter interaction

As the voter receives these two message (Message E, F), he /she can't decrypt message E as it encrypted with B-voting server secret key, so the only one that can decrypt it is the B-voting server itself . The voter can only decrypt message F as he/she already had the associated voter TGS session key from A ($SK_{V\text{-}TGS}$), so the decryption is performed $D(SK_{V\text{-}TGS}(\text{Message F}))$ and the voter can obtain now the voter B-voting server session key($SK_{V\text{-}B\text{-}VotingServer}$). The voter now will contact the B-Voting server and send two messages (Message E ,G). The **Ferguson E cash protocol had been modified to be used in E voting**, it had been combined with the modified Kerberos protocol at this step, in order to verify the whole voter identity and certificate ($Cert_v$):

(1) In addition to these two message (E ,G) that send to B-voting server. The voter will select two blind factors b_1 and b_2 and three random numbers $x_1, x_2 \in Z_{e'_{BV}}^*$ and $s \in Z_{e'_{BV}}^*$ and compute A ,A' ,B, w_1, w_2 as follows :

$$A = g_1^{u_v} g_2 \mod n_{BV} \dots\dots\dots (3.23)$$

$$A' = A^s \mod n_{BV} \dots\dots\dots (3.24)$$

$$B = g_1^{x_1} g_1^{x_2} \mod n_{BV} \dots\dots\dots (3.25)$$

$$w_1 = B b_1^{e'_{BV}} \mod n_{BV} \dots\dots\dots (3.26)$$

$$w_2 = (A' + B) b_2^{e_{BV}} \mod n_{BV} \dots\dots\dots (3.27)$$

Then, the voter send $\{Cert_v, A, w_1, w_2, t, ((A || w_1 || w_2 || t)^{d_v}) \mod n_v\}$ to B-voting Server

(2) As the B-voting Server receive this message, again it will verify the validity of the voter certificate, but at this time it won't contact a responder, rather it will contact a

PVID Authority database to check if this voter is legal to participate in voting and thus he /she has a certificate. Also the B-voting Server will contact an AS database, as AS verify the validity of the voter obtained certificate. After B-voting server verify the validity of the certificate, timestamp and value of A by using certificate, identity of the voter and public information. It also validates the signature $(A||w_1||w_2||t)^{d_v} \bmod n_v$. After passing all the verification, B-voting server will compute the following equation:

$$w_3 = A^{\frac{1}{e_{BV}}} \bmod n_{BV} \dots\dots\dots(3.28)$$

$$w_4 = w_1^{\frac{1}{e_{BV}}} \bmod n_{BV} \dots\dots\dots(3.29)$$

$$w_5 = w_2^{\frac{1}{e_{BV}}} \bmod n_{BV} \dots\dots\dots(3.30)$$

(3) Finally the message $\{(w_3, w_4 w_5)^{e_v} \bmod n_{BV}\}$ is sent to V

(4) Decrypting the received value, V will get access to the signature of B-Voting server on A and blinded signature of B-voting server on B and $A' + B$. Voter compute the signature of B-voting server on A' , B and $A' + B$ as follows :

$$s_1 = w_3^s \bmod n_{BV} = A'^{\frac{1}{e_{BV}}} \dots\dots\dots(3.31)$$

$$s_2 = \frac{w_4}{b_1} \bmod n_{BV} = B^{\frac{1}{e_{BV}}} \dots\dots\dots(3.32)$$

$$s_3 = \frac{w_5}{b_2} \bmod n_{BV} = (A' + B)^{\frac{1}{e_{BV}}} \dots\dots\dots(3.33)$$

On the other side the Kerberos protocol is still operate behind the above steps in Ferguson protocol, two messages will be sent to the B-Voting server by the voter (see Figure 3.20):

-Message E : Same one in the previous step

-Message G: Is an authenticator that consist of the *PVID-list* and a timestamp (e.g. may indicated the current date and time) encrypted with voting B-voting server session key from message F ($SK_{V-B-VotingServer}$)

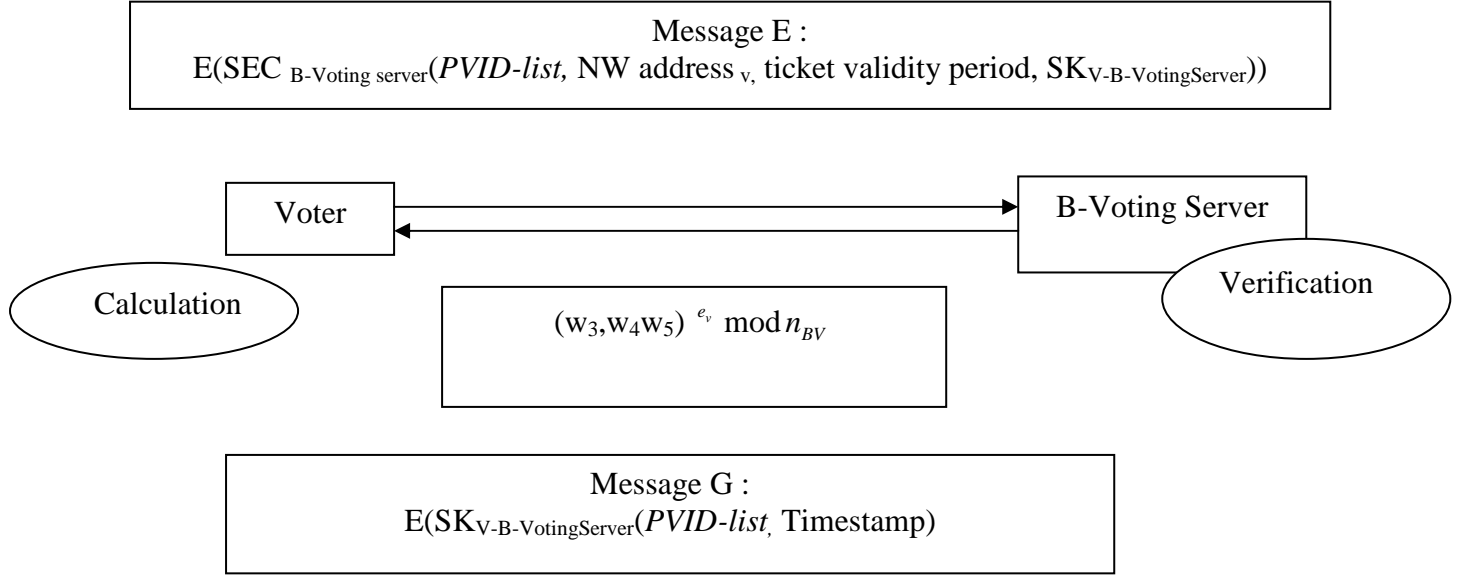


Figure 3.20: Voter-B-voting server interaction

As the B-Voting Server receive these two messages (Message E, G), it can decrypt message E, using the B-voting server secret key and thus retrieve the associated information from message E: $D(\text{SEC}_{\text{B-Voting server}}(\text{Message E}))$. As the B-Voting server get $\text{SK}_{\text{V-B-VotingServer}}$ from message E decryption, the voter can decrypt message G as the following $D(\text{SK}_{\text{V-B-VotingServer}}(\text{Message G}))$. As B-voting server decrypt these two messages (Message E, G), it can make a match between them as the following \rightarrow if $(\text{PVID-list from Message E} == \text{PVID-list from Message G} \ \&\& \ \text{Timestamp.Message G} \leq \text{Ticket Validity period.Message E})$. If all above steps in both Ferguson and Kerberos were successfully passed, the B-Voting server will confirm the voter's true identity, so the B-Voting server now will send a message H (see Figure 3.21).

-Message H :

It contains the authenticated ticket and the timestamp+1 (timestamp that found in message G in the previous step increment by one) encrypted with the voter and B-voting server session key ($\text{SK}_{\text{V-B-VotingServer}}$). Additionally, it will send a $[E(\text{SK}_{\text{V-B-VotingServer}}(\text{PK}_{\text{voting}}) + h(\text{PK}_{\text{voting}}))]$, so the voter can verify the received $\text{PK}_{\text{voting}}$ at step(1) from PVID with the $\text{PK}_{\text{voting}}$ received here. Also verify the received $\text{PK}_{\text{voting}}$ by computing the same hash function for it.

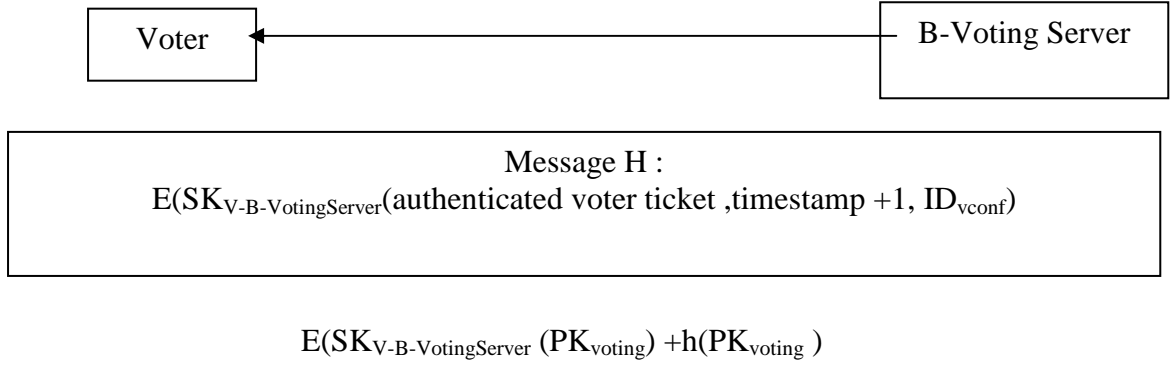


Figure 3.21: B-voting server –voter interaction

As the voter receive message H, he/she can decrypt using $SK_{V-B-VotingServer}$ and check if the timestamp (voter send in message G) is updated by one. Furthermore, as the voter has the PK_{voting} from the beginning, it can compare it with the received PK_{voting} from B-voting server, either if the same or not. On other side, the voter will decrypt $D(SK_{V-B-VotingServer} (PK_{voting}) +h(PK_{voting}))$, and thus get the $(PK_{voting}) +h(PK_{voting})$. The voter is now able to calculate the same hash function for the decrypted PK_{voting} and make such a match. By this the eligible voter is get ready for the voting phase.

3.2 Voting Phase :

The voting phase consists of two sub phases, obtaining a dynamic ballot that hold the candidate from which the voter can choose his/her vote, and a voting submission phase itself at which the voter choose his/her candidate and blinded it in order to be sign by administrator (collude administrator will be solved) and later to be counted.

3.2.1 Ballot obtaining phase :

In this phase two entities will be involved the electronic ballot generator (EBG) and the key generator (KG), first the voter will compute the value of d , r_1 , r_2 using the following equations:

$$d = H(A', B, s_1, s_2, s_3, b_2(\text{candidate}), \text{nonce}) \bmod n_{EBG} \dots\dots\dots (3.34)$$

$$r_1 = d_{u,s} + x_1 \bmod n_{EBG} \dots\dots\dots (3.35)$$

$$r_2 = d_s + x_2 \bmod n_{EBG} \dots\dots\dots (3.36)$$

Then, the voter will prepare a request to obtain a dynamic ballot, the voter creates session public-private key pairs (α_x, μ_x) for electronic ballot generator and (α_y, μ_y) for key generator. He/she employs these keys in order to obtain a dynamic ballot. Voter encrypts α_y and election data produces $E(\text{PU}_{KG}(\alpha_y, \text{ElectionData}))$. The important of election data that it makes the message easily identified by the key generator.

$M_1 = E(\text{PU}_{EBG}(\text{PVID}_1, E(\text{PU}_{KG}(\alpha_y, \text{ElectionData})), \alpha_x, d, r_1, r_2, A', B, s_1, s_2, s_3), \text{PK}_{\text{voting}})$. The voter will send M_1 to electronic ballot generator EBG (see Figure 3.22)

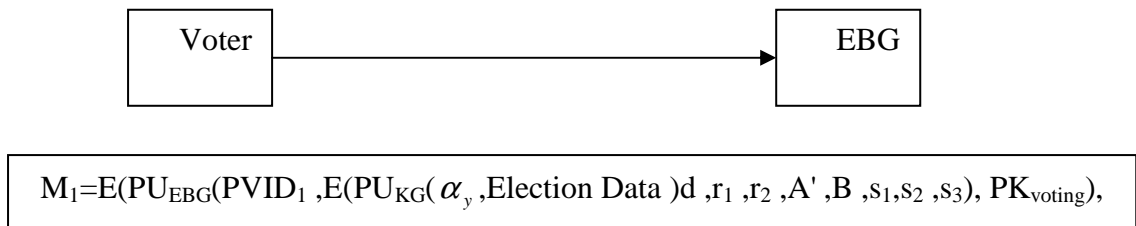


Figure 3.22: Voter-EBG interaction 1

As the M_1 is received, the EBG will decrypt it using its own private key $(D(\text{PR}_{EBG}(M_1)))$ and thus verifying the PVID authority signature on PVID_1 , the signatures of s_1, s_2, s_3 , computing the hash value for the received voting public key then compare it with the received one from the commissioner and the following

equation $g_1^{r_1} g_2^{r_2} = A'^d B \mod n_{EBG}$ to ensure that no items have been forged in the protocol. Noted that it can't deal with the concealed message as it encrypt with the key generator public key (PU_{kg}), so the only one can decrypt is the KG itself, any discover of a forgery attempt, EBG will discard the message. Otherwise, it sign message and generate M_2 . Here is provided another way fro eliminate double voting as the values of $d, r_1, r_2, A', B, s_1, s_2, s_3$ will be stored in EBG database. If these parameters appeared twice as the following explain: {ElectionData, $d, r_1, r_2, A', B, s_1, s_2, s_3$ } and another received message with same values {ElectionData, $d', r_1', r_2', A', B, s_1, s_2, s_3$ } it will be easily detected by using the relation between r_1, r_2, d and consequently between r_1', r_2', d' it will compute the identity of the voter as by the following equations :

$$u_v = \frac{r_1 - r_1'}{r_2 - r_2'} \mod n_{EBG} \dots\dots\dots(3.37)$$

By this the eligible voter only vote once and any attempt for voting again will be easily detected. Now the EBG will send $M_2 = E(PU_{KG}(PR_{EBG}(PU_{KG}(\alpha_y, \text{Election Data}))))$ to the KG (see Figure 3.23).

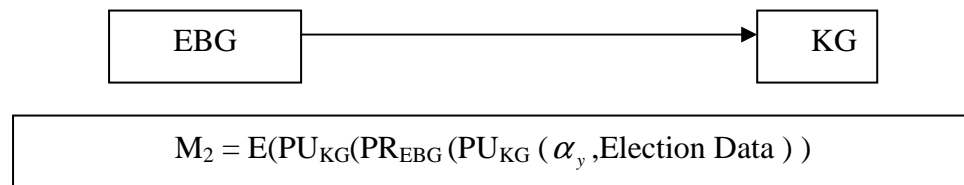


Figure 3.23: Voter-EBG interaction 2

Key generator that will decrypt M_2 ($D(PR_{KG}(M_2))$) and verify from the electronic Ballot generator (EBG) signature. After such verification performed the KG will send an encrypted message (M_3) to the EBG that contains a hash for PK_{voting} ($H(PK_{voting}) \rightarrow E(PR_{KG}(H(PK_{voting})))$) (step 1 in Figure 3.24).

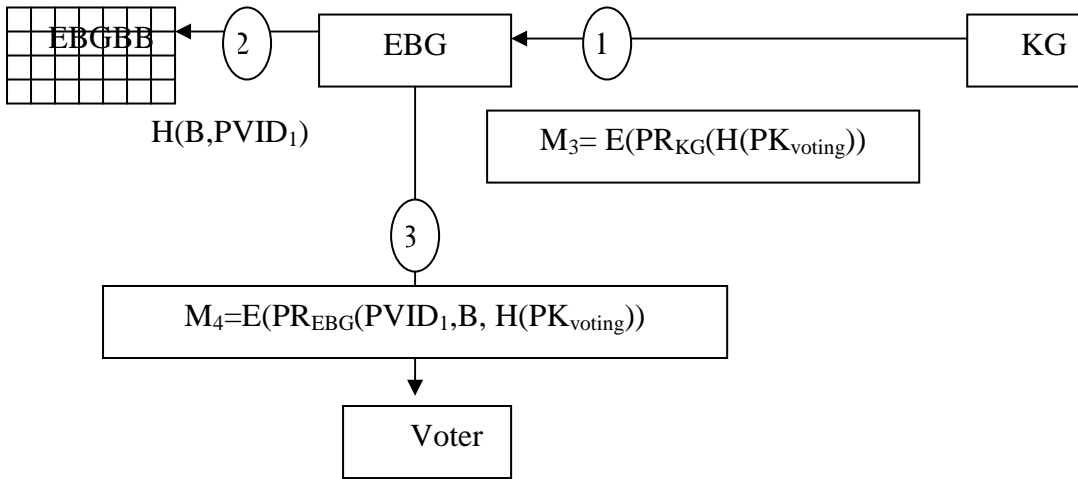


Figure 3.24: Voter-EBG-KG interaction

As a response for receiving M_3 , EBG will verify the message M_3 and create a voter dynamic ballot B by using a ballot generation algorithm relying on a random number generator function. Dynamic Ballot B orders candidates randomly. The EBG will send a hashed ballot B and linked it to the voters $PVID_1$ for verification purpose. So the voter can check if the ballot that he/she received is the same that the EBG publish on his/her own bulletin board. This will assure the voter and any higher authority (e.g. commissioner) that the EBG doesn't modify ballot. EBG will also save B , $PVID_1$ for each voter and sends them to counter after election periods ends. As shown in Figure 3.24, step 2. After that, the EBG will send to the voter M_4 that consists of the dynamic ballot B , $H(PK_{voting})$ that received from M_3 and a signed $PVID$ with the PR_{EBG} (as the Figure 3.24 shows). As soon as the voter receives M_4 , he/she can verify a gain the PK_{voting} as the voter has such a key he/she can calculate the hash for it and compare it with the received hash.

3.2.2 Voting Submission phase

The voter will be able to choose his/her vote from such a dynamic ballot and blinded the vote with b_2 , such that $\gcd(n, b_2) = 1$ ($b_2^{Y(i)}(\text{vote})$) and encrypted the vote with PK_{voting} and submit to administrators for signing purpose. Remember that the administrator public key is denoted as the following (Y_i) and the notation used for the administrator public key is Y_i where (Y_i, n) is the public key of the administrator and u is the private key of administrator where (X_i, n) is the private key of administrator.

This ballot will be sent to administrators for signing purpose. In order to avoid the collude administrators problem, a signature should had $t > n/2$, this can be done by applying the PVSS protocol based on a threshold signature², also the ballot will contain the authenticate voter ticket that was generated previously (Kerberos Authentication protocol), So the administrator will never sign a ballot for an ineligible voter. Consequently, DoS attack will be eliminated the counter buffer will never be filled with a garbage votes.

In order for the voter to obtain a sign for his/her vote, he/she should send an encrypted message, with the dealer public key (PU_{dealer}), to the dealer. This message consists of (the voter authenticated ticket, $PVID_1$) as the following: $E(PU_{\text{dealer}}(\text{authenticated voter ticket}, PVID_1))$, see Figure 3.25.

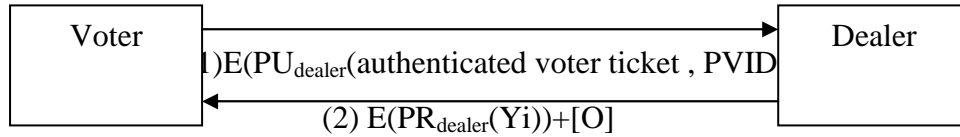


Figure 3.25: voter-EBG interaction

As the dealer receives this message, (see Figure 3.25, step 1) it will be decrypted, using dealer private key and obtaining the associated information. Thus observing the voter authenticated ticket and realizing that the voter is authenticated, also verifying the $PVID_1$ by checking the $PVID$ authority signature. As such verification is successfully completed, the dealer will send to the voter the generated administrators public key (Y_i), it will be illustrated later in the PVSS scheme, for

² We describe the construction for a (t, n) -threshold access structure, but it can be applied to any usual access structure for which a linear secret sharing scheme exists

verifying the administrators signature on the ballot (blinded signed vote), the (Y_i) will be encrypted with the dealer private key as shown in Figure 3.25, step 2, as soon as it receive by the voter a verification by voter is done via a dealer public key. It is noted that any authority server involved in the proposed scheme can obtain an administrators public key by contact a dealer server.

After the voter has the administrators' public key (Y_i) , the voter will send a ballot containing the blinded vote with Y_i : $B=\{(b_2^{Y(i)}(\text{vote}))\}$ via a dealer to administrators for signing purpose, depending on the blind signature the following equation shown that

$$x=b_2^{Y(i)}(\text{vote}) \bmod n \dots\dots\dots 3.38$$

Step (1) from Figure 26), in order to avoid a collude administrators; a PVSS based on a threshold signature (review the literature) will be applied as will be illustrated later (review the literature for general view for secret sharing based on a threshold cryptosystem).

In order to avoid the collude administrator problem, the PVSS will be applied, so that no one of the administrators can collude with others to alter voting results or send a forgery vote. Let G_q denote a group of prime order q , such that computing discrete logarithms in this group is infeasible. Let g, G denote independently selected generators of G_q , hence no party knows the discrete log of g with respect to G . The problem of efficiently sharing a random value from G_q is solved. The dealer will achieve this by first selecting $s \in_R$. So and then distributing shares of the secret $S = Gs$. This approach allows us to keep the required proofs simple and efficient.

By using the protocol introduced by (Chaum & Pedersen, 1993) as a sub protocol to prove that $\log_{g_1} h_1 = \log_{g_2} h_2$, for generators $g_1, h_1, g_2, h_2 \in G_q$. This can be denoted as DLEQ (g_1, h_1, g_2, h_2) and it consists of the following steps, where the prover knows α such that $h_1 = g_1^\alpha$ and $h_2 = g_2^\alpha$

1. The prover sends $a_1 = g_1^w$ and $a_2 = g_2^w$ to the verifier, with $w \in_R \mathbb{Z}_q$.
2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
3. The prover responds with $r = w - \alpha c \pmod{q}$.
4. The verifier checks that $a_1 = g_1^r h_1^c$ and $a_2 = g_2^r h_2^c$

✓ **Initialization:**

The group G_q and the generators g, G are selected using an appropriate public procedure. Participant P_i generates a private key $x_i \in {}_R Z_q^*$ and registers $y_i = G^{x_i}$ as its public key.

✓ **Distribution:**

(1) Distribution of the shares :

Suppose that the dealer wishes to distribute a secret (to prevent administrators to collude) among participants P_1, \dots, P_n ³. The dealer picks a random polynomial p of degree at most $t - 1$ with coefficients in Z_q :

$$p(x) = \sum_{j=0}^{t-1} \alpha_j x^j \dots\dots\dots(3.39)$$

, and sets $s = \alpha_0$. The dealer keeps this polynomial secret but publishes the related commitments $C_j = g^{\alpha_j}$, for $0 \leq j < t$. The dealer also publishes the encrypted shares $Y_i = y_i^{p(i)}$, for $1 \leq i \leq n$, using the public keys of the administrators. Finally, let $X_i = X_i = \prod_{j=0}^{t-1} C_j^{i^j}$. The dealer will show that the encrypted shares are consistent by producing a proof of knowledge of the unique $p(i)$, $1 \leq i \leq n$, satisfying:

$$X_i = g^{p(i)}, Y_i = y_i^{p(i)} \dots\dots\dots(3.40)(\text{Stadler,1996})$$

The non-interactive proof is the n -fold parallel composition of the protocols for DLEQ (g, X_i, y_i, Y_i). Applying Fiat-Shamir's technique, the challenge c for the protocol is computed as a cryptographic hash of X_i, Y_i, a_{1i}, a_{2i} where $1 \leq i \leq n$. The proof consists of the common challenge c and the n responses r_i .

2. Verification of the shares: The verifier computes $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ from the C_j values.

Using y_i, X_i, Y_i, r_i , $1 \leq i \leq n$ and c as input, the verifier computes a_{1i}, a_{2i} as

$$a_{1i} = g^{r_i} X_i^c \dots\dots\dots(3.41)(\text{Stadler,1996})$$

$$a_{2i} = y_i^{r_i} Y_i^c \dots\dots\dots(3.42)(\text{Stadler, 1996})$$

and checks that the hash of X_i, Y_i, a_{1i}, a_{2i} , $1 \leq i \leq n$, matches c .

³ Participants equivalent to players ,administrators

✓ **Reconstruction:** The protocol consists of two steps:

1. Decryption of the shares. Using its private key x_i , each administrator finds the share

$S_i = G^{p(i)}$ from Y_i by computing $S_i = Y_i^{\frac{1}{x_i}}$. They publish S_i plus a proof that the value S_i is a correct decryption of Y_i . To this end it suffices to prove knowledge of an α such that $y_i = G^\alpha$ and $Y_i = S_i^\alpha$, which is accomplished by the non-interactive version of the protocol DLEQ (G, y_i, S_i, Y_i).

2. Pooling the shares: Suppose that participants P_i produce correct values for S_i , for $i = 1, \dots, t$. The secret G^s is obtained by Lagrange interpolation:

$$\prod_{i=1}^t S_i^{\lambda_i} = \prod_{i=1}^t (G^{p(i)})^{\lambda_i} = G^{\sum_{i=1}^t p(i)\lambda_i} = G^{p(0)} = G^s, \quad \dots\dots(3.43) \text{ (Stadler, 1996)}$$

where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$, Is a Lagrange coefficient

Note that the administrators do not need nor learn the values of the exponents $p(i)$. Only the related values $S_i = G^{p(i)}$ are required to complete the reconstruction of the secret value $S = G^s$. Also, note that participant P_i does not expose its private key x_i ; consequently participant P_i can use its key pair in several runs of the PVSS scheme. Clearly, the scheme is homomorphic. For example, given the dealer's output for secrets G^{s1} and G^{s2} , the combined secret G^{s1+s2} can be obtained by applying the reconstruction protocol to the combined encrypted shares $Y_{i1}Y_{i2}$. As the participants (administrators) share a secret securely without any collision using PVSS based on a threshold signature, this share secret can be used to sign the voter vote $B\{(b_2^{Y(i)}(\text{vote}))\}_{S_i}$, within a blind signature the following formula :

$$t = x^{s_i} \bmod n \quad \dots\dots\dots(3.44)$$

(See Figure 2.26, step 2). Anonymously this signed administrators ballot will be send to the voter through a dealer. (See Figure 2.26, step 4), so that the voter can have the administrators signed vote without the administrator knowing the signed vote (depending on blind signature scheme, and as the following equation indicated :

From(3.44) and as $x = b_2^{Y(i)}(\text{vote})$

$$t = (b_2^{Y(i)}(\text{vote}))^{s_i} \bmod n$$

$$t = b_2^{Y_i s_i}(\text{vote})^{s_i} \bmod n$$

$$t = b_2(\text{vote})^{s_i} \bmod n$$

taking b_2^{-1} to both sides

$$b_2^{-1} t = (\text{vote})^{s_i} \bmod n$$

$$s = (\text{vote})^{s_i} \bmod n \dots \dots \dots (3.45)$$

Where s is the vote v signed by using the administrators shared secret (private key) preventing collude administrators and that no administrator can know the voted signed. Furthermore, a dealer will publish a hash for the signed blindly ballot on the bulletin board so that the dealer will verify any signing request if it had already signed before. If not, it will be forwarded to the administrators for signing. After such publication on the bulletin board, the voter can verify the received signed ballot by computing the same hash for it and compare it with the one associated on the bulletin board (See Figure 2.26, step 3).

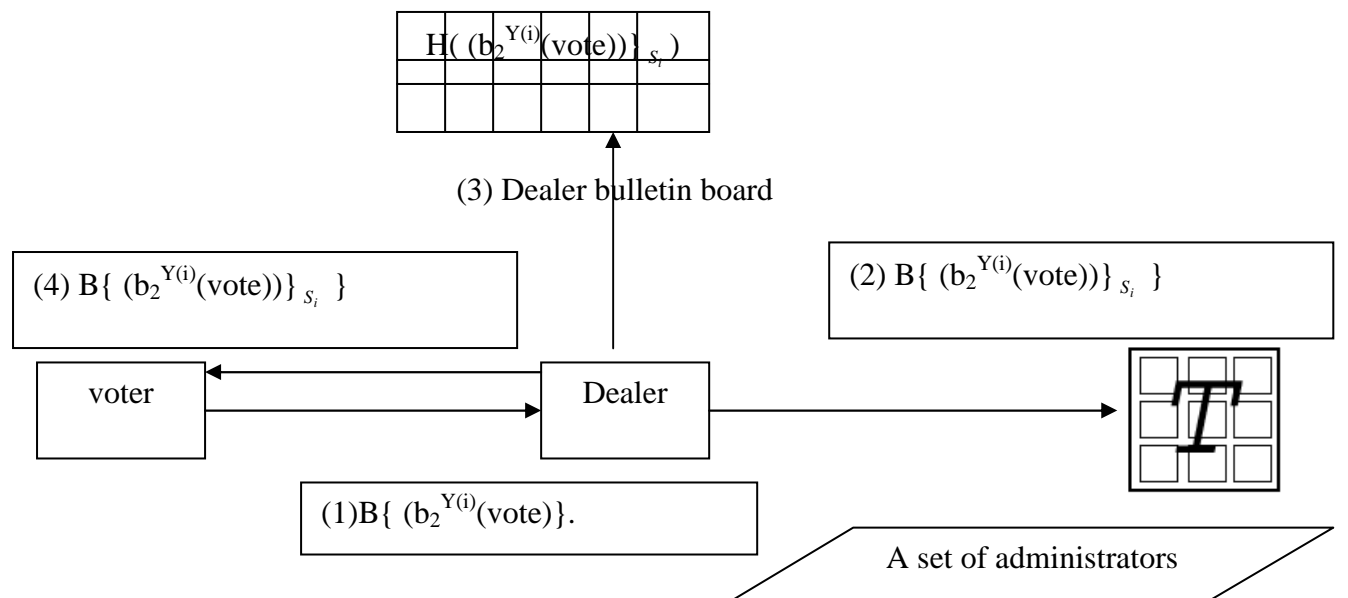


Figure 3.26: Voter-Dealer-Administrators interaction for ballot signing

3.3 Counting Phase :

Now, after the voter gets his/her signed vote $((vote)^{s_i})$, the voter will encrypt it using the voting public key issued by the commissioner (PK_{voting}) within a Triple-DES cryptographic algorithm (see the proposed scheme analysis), so the ballot will contain encrypted signed vote $B = \{E(PK_{\text{voting}}((vote)^{s_i}))\} = (N, h)^4$ accompanied with the $PVID_1$. The voter will commit to the ballot using a trapdoor commitment scheme (Review the literature) and sending it with a $PVID_1$ anonymously to the counter for counting purpose as will be seen later. For the ballot commitment purpose the voter will generate $B_c = C(r_1, c_j) || B_b = c(r_1, b_t)$ with a trap door commitment scheme in the message commitment phase, $B_c = C(r_1, c_j) = h^{r_1} (1 + c_j N) \bmod N^2$ and send the committed ballot (B_c, r_1, c_j) anonymously to the counter for counting purpose accompanied with a $PVID_1$, for individual verifiability property purpose (see the Analysis chapter).

Another way is provided here to prevent a double voting: By using a trapdoor commitment scheme the counter can eliminate the duplicate ballot, so the counter ignores the duplicate one, and counts it only once (in case a voter try to make another vote as the following illustrate

$B_c = C(r_1, c_j)$	$B_b = c(r_1, b_t)$
---------------------	---------------------

Another benefit gained from applying the trapdoor commitment scheme is that the possibility for counters to collude will be low due to the collision finding property related to the trapdoor commitment scheme. After the voter's committed to the ballot using trap door commitment scheme:

- The voter will anonymously send $PVID_1$ (same one accompanied with the committed ballot that send to the counter) with a trap door secret, r , to the commissioner. These will be stored at the commissioner data base.
- The voter generates a digest of the committed ballot ($H(CB)$) and publish to the voter bulletin board, via generating a random blinding factor to apply to the digest. Any counter involved in ballot counting operation can verify such ballot by

⁴ Depending on the applied trapdoor commitment scheme notation (review Literature)

calculating the same hash for the received committed ballot and compare it with the one in the voter bulletin board. The received committed ballot and PVID₁ will be stored in the counter database.

- Sending committed ballot and PVID₁ anonymously for counting purpose (as it received, it will be stored in the counter database).

After the end of the election the commissioner will publish the voting private key (PR_{voting}) for ballot decryption purpose and r value associated with each PVID. For each PVID associated at counter side, the corresponding value r will be used to retrieve the ballot (without the commitment). Mainly the counting process involve the following major steps:

- ✓ Removing repeated votes, which are the one with the same bit commitment.
- ✓ Mixing authority mixes $B_c = C(r_l, c_j) \parallel B_b = c(r_l, b_t)$.

Obtaining $\phi[B_c = C(r_l, c_j)] \parallel \phi[B_b = c(r_l, b_t)]$ as indicated in Table 3.2.

Table 3.2: Proof obtaining values

$\phi[B_c = C(r_l, c_j)]$	proof	c_j
$\phi[B_b = C(r_l, b_t)]$	proof	b_t

- ✓ As the counter received values ($r + PVID_k$) from commissioner, the received PVID_k would be compared with others stored in the counter database until a match is found. As a match found, the r value associated with PVID_k will be used to remove commitment on the ballot and then decrypt the received ballot using $(PR_{\text{voting}}) = (p, q)^5$ as shown in Table 3.2, and get the associated (c_j, b_t) . By this, the researcher notices that there isn't any interaction from voter as the associated trapdoor kept secret, but the voter provided necessary information (e.g. encrypted value of r_l) for a commissioner to finally let the counter to open the commitment ballot.
- ✓ Verifying the administrator signature on the vote by contact a dealer and obtain the administrators public key(Y_i).
- ✓ Counting the remaining votes.

⁵ Depending on the applied trapdoor commitment scheme notation (review Literature)

- ✓ In the announcement bulletin board the votes will be published. Each vote with the associated PVID for that vote. By this voters can assure that there votes were really counted. (Individual verifiability is achieved).
- ✓ publish the final results.

Noted that it is better for counter to publish a hash for the vote on the bulletin board so the voter can verify that his/her vote had been counted correctly by computing the same hash for his/her vote and compare it with the one publish on the bulletin board .

By this the researcher guarantees in the secure proposed scheme that the counter buffer will never be filled with garbage votes as only authorized voter are permitted to vote, any double voting issued by authorized voter will be easily detected as more than one security mechanism was applied and the collude administrator problem easily solved by applying a PVSS based on a threshold signature.

Chapter Four

Analysis and Discussion

In the previous chapter, the proposed scheme is explained in detail. This chapter provides an analysis for it. The researcher analyzes the proposed scheme from two points of view. The first one details how the proposed scheme meets the security requirements that any E voting scheme or system should meet. The second one involves a simulation result using a dizzy simulator and discussion for the results obtained.

4.1 Security Requirements

In this section, the researcher proves the correctness of the proposed scheme to fulfill the claimed properties (review the literature). In order to do that, first: the researcher assumes the following:

Assumption One: Factorization of large numbers is a hard problem.

Assumption Two: The cryptographic algorithms used are hard to break .The mainly involved are:

- (1) RSA, for producing and checking blind and non blind signatures
- (2)Triple-DES, used to encrypt the vote using a PK_{voting}
- (3)SHA-1,for all required hash (digest) component.

Assumption Three: Anonymizers and counters are honest (collude problem can't arise between them) and a secure communication is available

Assumption Four: The servers, voter's computers and all communication gateways aren't vulnerable to attacks such as infection or Trojan horses or viruses.

Assumption Five: Elliptic curve is secure.

Second: The researcher adopt the same analyzing method that used in Dyna vote analysis (Cetinkaya & Koc, (2009)). Next, the researcher details how these requirements were met in the proposed scheme.

4.1.1 A Method to Analyze Voting Systems

While E voting has been studied for the past two decades, research on analyzing voting systems has begun recently. In this section, a method to analyze voting scheme with respect to E voting security requirements is adopted. This method helps to evaluate, as well as compare, the voting protocols and it is not protocol specific. In order to define a voting protocol VP , let:

- $E = \{e_1, e_2, e_3 \dots e_q\}$ be the set of all eligible voters where q is the number of eligible voters.
- $A = \{a_1, a_2, a_3 \dots a_n\}$ be the set of voters that performed a voting process where a_i is any voter and n is the number of voting attempts.
- $B = \{b_1, b_2, b_3 \dots b_n\}$ be the set of votes where b_i is the vote of voter a_i .
- $D = \{d_1, d_2, d_3 \dots d_n\}$ be the set of transactions in voting processes where d_i denotes all transactions of voter a_i during the voting process.
- $V = \{v_1, v_2, v_3 \dots v_m\}$ be the set of all valid votes (including all data) where m is the number of valid votes, $V \subseteq B$ and $m \leq n$.
- $W = \{w_1, w_2, w_3 \dots w_m\}$ be the set of published data at the end of the election, w_i denotes the published data for each valid vote v_i and $w_i \subseteq v_i$.
- $C = \{c_1, c_2, c_3 \dots c_k\}$ be the set of all candidates.
- $F_{bv}: B \rightarrow V, f_{bv}(b_i) = v_j$ matches each b_i to a v_j if b_i is a valid vote.
- $F_{ae}: A \rightarrow E, f_{ae}(a_i) = e_j$ matches each a_i to an e_j if a_i is an eligible voter.
- $F_{vc}: V \rightarrow C, f_{vc}(v_i) = c_j$ matches each valid vote to an actual candidate.
- $S = \{s_1, s_2, s_3 \dots s_h\}$ be the set of all eavesdroppers.
- $T =$

$$\{(c_1 \sum_{i=1}^m add(f_{vc}(v_i), c_1)), (c_2, \sum_{i=1}^m add(f_{vc}(v_i), c_2)) \dots (c_k, \sum_{i=1}^m add(f_{vc}(v_i), c_k))\}$$

be the final tally.

Note that if any recasting occurs then it is handled as a new voting process, so it can be $n \geq q$. If recasting is not allowed, then it should be $n \leq q$. Besides, D does not require to be hidden.

4.1.2 Formal Definitions of E-voting Security Requirements

In order to guarantee that E voting security requirements were met in the proposed scheme, the researcher evaluates the proposed scheme by first introducing a formal definition for these requirements (Cetinkaya & Koc, (2009)), then mathematically prove each of them. Finally, for each requirement a checklist items is given below for a requirement brief summary proof.

Lemma1 Privacy (Voter-Vote relationship cannot be revealed):
If $\forall d \in D \forall v \in V \forall e \in E [\neg(\exists f(S, W, d, v) = e)]$ for a voting scheme VS, then VS satisfies privacy.

Proof ⁶ : This requirement is met by applying a PVID scheme that relies on the unlinkability between voter's pseudo identity and real identity. In order to prove any relation between them, the random number used to create blinded message should be known. Otherwise, adversary should break RSA cryptosystem since PVID scheme uses blind signature based on RSA public key cryptosystem, which is infeasible. The random number is generated by voter and nobody knows it.

Frankly Speaking, after the voter obtains PVID list, the voter no more use his/her RegID, thus no adversary, including all authorities can find a function f such that $\forall v \in V \forall e \in E [\exists f(S, W, D, v) = e]$ so nobody can break the voter-vote unlinkability.

Additionally, by relying on the blind signature and according to its definition ,there is no function f satisfying $\forall v \in V \forall e \in E [\exists f(p) = e]$ in the proposed scheme by this the researcher guarantee that all votes will kept secret due to the blindness property under blind signature and thus no participant other than a voter should be able to determine the value of the vote cast by that voter as the voter sign his/her blinded vote without the sign authority (administrators) know the actual vote (see Table 4.1 that summarize the case related to privacy proof).

⁶ the proof refer to show how the proposed scheme satisfy the associated lemma

Table 4.1: Privacy requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Privacy	Voter-vote unlinkability	yes	-	-	Applying PVID scheme + Blind signature protocol	
	Voter-voteIP untraceability	yes	-	-	There is no point in trying to trace the voter IP since nobody can guarantee whether or not the voter accesses over a dynamic IP, he uses the voting pool or any other public network, and he employs any IP anonymizer application. In case of a voter having a static IP and not taking any care about it, then IP untraceability may fail if authorities corrupt.	none of the authorities keeps IP of the voters and releases them

Lemma 2 Eligibility (Each vote counted in the tally should be cast by an eligible voter): let $f : V \rightarrow B, f(v_i) = b_j$ and $g : B \rightarrow A, g(b_j) = a_j$. If $\forall v \in V [f_{ae}(g(f(v))) \in E]$ for a voting scheme VS, then VS satisfies eligibility.

Proof: By relying on the Kerberos authentication protocol infrastructure, the researcher guarantees that only authorized voters are permitted to vote by the generation of the issued voter ticket.

(1) Also the voter can't forge such a ticket without any detection

Proof (1): it can be proved by a contradiction. Let us suppose that a voter can forge the ticket. This means that the forged ticket is provided by changing in values of one of the signed amount $s_1 = \text{sign}_{BV}(A')$, $s_2 = \text{sign}_{BV}(B)$, $s_3 = \text{sign}_{BV}(A'+B)$. As the value of s_3 depend on the two previous value of s_1 and s_2 , changing the value of s_3 is impossible. As well as the value of B is optimal, B forging isn't valuable. So forging a ticket without detection is impossible.

(2) It becomes impossible to forge an extra ticket to vote with

Proof(2): This requires a forgery of the PVID-list signature which is impossible as the PVID authority issues blind signature on voters blinded ID too after checking against country election registration laws (e.g. above 18 years old). Let prove by a contradiction method too, assuming there exit a function $f : P \rightarrow E, f(p_i) = e_i$ that known only by the voter.

Then the proposed scheme satisfies $\forall v \in V [\exists ! e \in E \mid f(p) = e]$. Furthermore, depending on the prove in (1), the voter alone is unable to forge A. However, if voter colludes together for such extra ticket forgery, the forgery one is identified by dealer in the voting stage as a case of double voting. Finally the issued PVID authority certificate will never be forging due to the additional entity (responder) that verifies the certificate. (see Table 4.2 that summarize the case related to Eligibility proof).

Table 4.2 :Eligibility requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Eligibility	Eligible voters can vote	yes	-	-	Kerberos authentication protocol with an authenticated ticket + <i>PVID-list</i> signature	
	Ineligible voters cannot vote	yes	-	-	As PVID checking against voter ,a blinded voter identities will never be signed for ineligible voter +Kerberos authentication ticket guarantee only authorized voters were vote ,can't be forged (see proof(1) ,(2))	PVID is a trusted authority

Lemma 3 Uniqueness (There should be at most one valid vote for each eligible voter in the final tally). Let:

$$f : V \rightarrow B, f(v_i) = b_j \text{ and } g : B \rightarrow A, g(b_j) = a_j$$

$$\text{If } \forall v_i \in V \forall v_j \in V [f_{ae}(g(f(v_i))) = f_{ae}(g(f(v_j))) = f_{ae}(g(f(v_j))) \leftrightarrow i = j]$$

for a voting scheme VS , then VS satisfies uniqueness.

Proof :

- (1) Using a trapdoor commitment scheme a voter's ballot is unique ,any attempt for double voting will be discarded by a counter, as the voter generate $B_c = C(r_1, c_j) \| B_b = c(r_1, b_t)$ with a trap door commitment scheme in the message commitment phase, $B_c = C(r_1, c_j) = h^{r_1} (1 + c_j N) \bmod N^2$ and send the committed ballot (B_c, r_1, c_j) anonymously to the counter for counting purpose and if the counter observe during proof $B_c = C(r_1, c_j) \| B_b = c(r_1, b_t)$, one will be discarded.
- (2) Since the *PVID-list* values (e.g. $PVID_1$ differ from $PVID_2$ and both are unique in the same list) and can be verified using the PVID authority public key ,there is only one vote counted for each voter. There exist such a function $f : V \rightarrow P, f(v_i) = p_j$,so uniqueness is satisfied as there is a true value for $\forall v_i \in V \forall v_j \in V [f(v_i) = f(v_j) \leftrightarrow i = j]$
- (3) Depending on the uniqueness of the PVID authority issued voter certificate ($Cert_v$) and the Kerberos authenticated voter ticket which obtain only once for an authorized voters and permitted them to participate in election during the specified election period, under the assumption that the PVID authority is trusted and thus can't forge certificate, if it forged a responder will verify it (responder and PVID authority can't collude, disjoint set with direct communication only) (see Table 4.3 that summarize the case related to privacy proof).

Table 4.3: Uniqueness requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Uniqueness	At most one valid vote is counted for each eligible voter	yes	-	-	Applying Trapdoor commitment scheme	
	Each eligible voter has voted only once	yes	-	-	Kerberos authentication protocol (Ticket issued once) +	

Lemma 4 Fairness (During the election none of the votes can be matched to an actual vote). During the election, if $\forall b \in B[\neg(\exists c \in Cf(D, S, b) = c)]$ for a voting scheme VS , then VS satisfies fairness.

Proof: As the commissioner publish the PR_{voting} at the end of the election for votes counting purpose, no participant can gain any knowledge about the (partial) tally before the counting stage, the voter will commit to the ballot using a trapdoor commitment scheme after the voting stage is completed, so no one can gain any partial knowledge about the tally before the counting stage is completed. Thus, counter cannot obtain the partial result. None of the authorities send any data to counter during the election period; counter cannot start counting before the end of the election. (see Table 4.4 that summarize the case related to privacy proof).

Table 4.4: Fairness requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Fairness	Result is not published till the end of the election	yes	-	-	Till the commissioner publish the PR _{voting}	
	No one can guess the content of any cast vote	yes			Trapdoor commitment +PVID scheme(unlinkability) +blind signature	
	No one can gain any partial knowledge about the tally before the counting stage	yes			Trapdoor commitment scheme + commissioner publish the PR _{voting} at the end of the election	All of the authorities do not cooperate in order to get partial result of the election

Lemma 5 Receipt-freeness (Uncoercibility) (Voters cannot prove their votes and thus No coercer can figure out a voters vote by forcing him):
 If $\forall a \in A \forall v \in V [\neg(\exists f(D, W, a) = v)]$ for a voting scheme VS , then VS is receipt-free.

Proof:

By applying the trapdoor commitment scheme, a vote recasting due to the fault tolerant is possible. If someone coerces a voter, even by only being physically next to him, the voter will cast in a way the coercer influences. Later, he/she can change his/her vote, by recasting a new vote which will automatically discard the old one in the counting stage. Even if the voter records his/her voting activity, still he cannot convince the coercer of the content of his/her vote due to recasting. That is, practically it is not possible to coerce or vote buy, since nobody can know whether the current vote will be the final one or not ; due to the trapdoor commitment and as the voter can provide $C_{(fake\ r, fake\ C_j)}$ and can be verified but can't find that a C_{j-fake} is a fake credential by a coercer, at the same time in the counting stage the voter will send the committed ballot by applying a trapdoor commitment scheme and related to the property that the voter can also find collision: $B_c = C(r_1, C_j) = C(fake\ r_1, fake\ C_j)$ $\parallel B_B = C(r_1, B_t) = C(fake\ r_1, fake\ B_t)$ and provide $B_c = C(fake\ r_1, fake\ C_j)$ $\parallel B_B = C(fake\ r_1, fake\ B_t)$. As the coercer has the ability to monitor the communication between voter and counter through anonymizer and verify without finding that C_{j-fake} is a fake credential .Thus, there is no function f satisfying $\forall a \in A \forall v \in V [(\exists f(D, W, a) = v)]$ for the proposed scheme. In practice, there is no point in coercing either physically or socially. Therefore, uncoercibility is achieved. (See Table 4.5 that summarize the case related to privacy proof).

Table 4.5 : Receipt freeness(Uncoercibility) requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Receipt-freeness (Uncoercibility)	Nobody can force voter to vote in a particular way	yes	-	-	Applying Trapdoor commitment scheme	
	Coercer cannot receive any proof from the voter after voting	yes	-	-	Voter can change his/her vote ,and only one counted by the counter due to trapdoor commitment scheme	

Lemma 6 Accuracy (Each vote cast by an eligible voter should be counted correctly in the final tally, and any fraud should be detected):

Let

$$h : E \rightarrow A, h(e_i) = a_j; g : A \rightarrow B, g(a_j) = b_j; f : V \rightarrow B, f(v_i) = b_j \text{ and } g' : B \rightarrow A, g(b_j) = a_j \\ \text{if } \forall e \in E [f_{bv}(g(g(e))) \in V] \wedge \forall v \in V [f_{ae}(g'(f(v))) \in E]$$

for a voting scheme VS, then VS satisfies accuracy.

Proof:

As bulletin boards are employed, each authority has its own bulletin board and hash of all information related to voters vote is recorded publicly. Thus, any corruption on the side of authorities can be detected. Counter compute the hashed for received votes and compare it with the one hashed associated with the dealer bulletin board so any passive observer or organization can also check the consistency of the election by using the bulletin board. Any cast vote cannot be altered, deleted, invalidated or copied since the modification causes inconsistency with the bulletin boards.

Even if Electronic Ballot Generator, Key Generator, and Counter conspire together, they cannot add a new vote since they can't create fake PVIDs. PVID Authority can't issue fake PVIDs under the assumption that any PVID authority involved is honest and trusted. PVID scheme assures that $\forall a \in A [\exists ! p \in P \rightarrow f_{ae}(a) \in E]$, as all votes are kept secret during the voting process, and having partial knowledge about voting data is not enough to vote or to simulate voter. Additionally, under the Kerberos authentication protocol issued ticket that will guarantee that only authorized voters are permitted to vote. The dishonest voter cannot disrupt the voting; he/she have just right over his/her own vote, so he/she may only disrupt his/her vote. Even if he/she sends more than one vote, in this case, the last one is counted, and the proposed scheme offers more than one approach to detect double voting before revealing the actual votes, counter performs some checks, due to the trapdoor bit commitment scheme. Thus, accuracy is achieved.

Proof (1): Related to the accuracy property, the assumption that the ballot representation is correct is true.

As a voter has the ability to provide correct values for r_1 and r_2 with respect to d which could pass the verification of voting and ticket obtaining phase in the Kerberos authentication protocol stages (between voter and B-voting server), if and only if the representation of A' and B with respect to g_1 and g_2 is known, if supposed that a voter known the representation of A' and B with respect to g_1 and g_2 . Consequently, he/she knows the values of u, x_1 and x_2 , then he/she can compute the values of d, r_1, r_2 . Conversely, suppose that a voter doesn't know the representation of A' and B with respect to g_1 and g_2 . Then, he/she doesn't know anything about u, x_1 and x_2 . So, he/she can't provide valid values for d, s, r_1 and r_2 . (See Table 4.6 that summarize the case related to privacy proof).

Table 4.6: Accuracy requirement details

Main Requirement	Requirement Details	Satisfied	Not satisfied	Not applicable	How it is satisfied	Assumption
Accuracy	Ballot representation is correct	yes			see proof(1)+Ferguson e-cash protocol conversion	
	No valid votes are either modified or deleted	yes			As bulletin board applied	
	No invalid votes are added	yes			PVID scheme +Kerberos authentication protocol	
	Any single authority corruption is detected	yes			Due to the bulletin board	

Lemma 7 Individual Verifiability (Each eligible voter should be able to verify his/her vote counted correctly): If $\forall e \in E \forall w \in W \exists ! v \in V [\exists f(e, w) = v]$ for a voting scheme VS , then VS satisfies individual verifiability.

Proof : After the end of the election the commissioner will publish the voting private key (PR_{voting}) for ballot decryption purpose and r value associated with each PVID. For each PVID associated at counter side, the corresponding value r will be used to retrieve the ballot (without the commitment), as the counter received values ($r + PVID_k$) from commissioner, it will be compared with the received $PVID_k$ with others stored in the counter database until a match is found. As a match is found, the r value associated with $PVID_k$ will be used to remove commitment on the ballot and then decrypt the received ballot using (PR_{voting}). Finally, in the announcement bulletin board the votes will be published. Each vote is associated with the PVID for that vote. By observing the announcement bulletin board; voters can assure that their votes were really counted. Individual verifiability is achieved. (see Table 4.7 that summarize the case related to privacy proof).

Table4.7 Individual Verifiability requirement details

Main Requirement	Requirement Details	Satisfi ed	Not satisfied	Not applic -able	How it is satisfied	Assumption
Individual Verifiability	Voter can validate that the ballot is correct	yes	-	-	Applying Trapdoor commitment scheme+see proof(1)in accuracy property	
	Voter can validate that authorities response Correctly	yes	-	-	As bulletin board applied	
	Voter can validate that his/her vote is recorded correctly	yes	-	-	As bulletin board applied + Trapdoor commitment scheme	

Theorem 1: A voting scheme VS is a complete and secure protocol if and only if it satisfies *Lemma 1-7*.

As all lemmas from 1-7 were satisfied, the researcher can conclude that the proposed scheme is secure.

4.2 Simulation Results

4.2.1 Voting Stage –Collude Administrator problem

In order to avoid the collude administrators problem, a guarantee should be available in which the ballot is signed by greater than the half of the administrator number ($n \geq 2$) as proposed in the Evox-MA E voting protocol based on blind signature (DuRette,1999), but nothing achieved toward this issue. However, REVS suggested the using of a double signing authentication for ballot signing by administrators and uses a different password for each administrator for such a purpose. On the other hand, this may force the voter to remember all the passwords used, even if an algorithm developed for password generation. This will add more overhead on the voters' module and thus consume time and complicate the process (Joaquim, et.al (2002)).

By using PVSS the researcher overcomes the problems associated so far, and gain a more stability (steady) results than the REVS, as will be shown later. The researcher suggests using of a dizzy simulator (Ramsey, 2006), (see Figure 4.1) in fact, it is a chemical kinetics stochastic simulation software package written in Java. The researcher models the E voting scheme as sequences of reactions⁷ between the communicating entities, by this the researcher can easily use dizzy simulator and analyze the scheme.

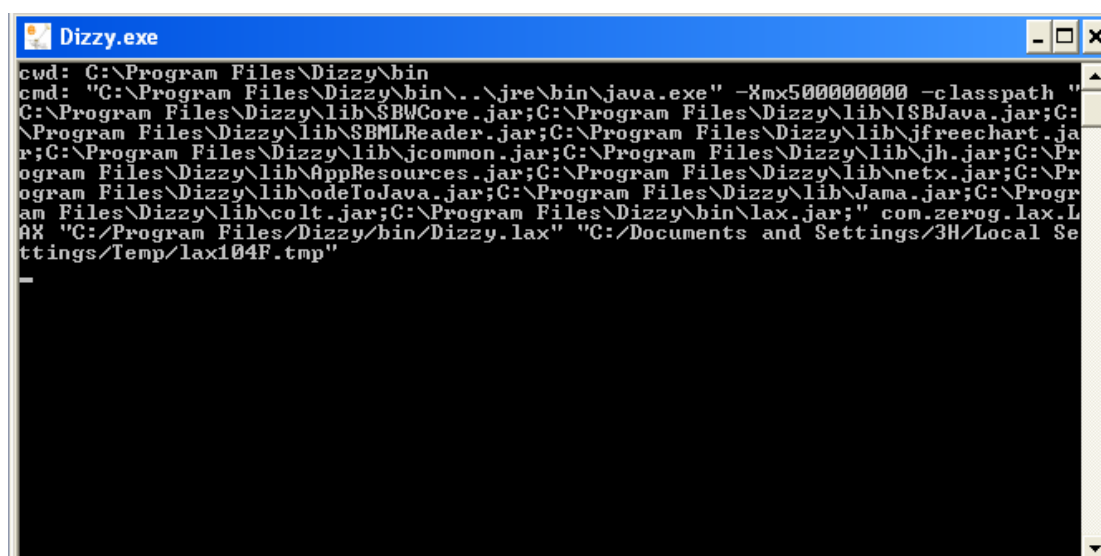


Figure 4.1: Dizzy simulator starting up

⁷ Chemical Model Definition Language (CMDL) that understood by Dizzy scripting engine.

In the voting stage, the communicating entities are the voters, dealers and administrators, the researcher suggest the use of PVSS to prevent the collude administrators problem (see Chapter Three), and the researcher presents the relationship between voters, Electionlistsize and administrators rate at this stage as a reaction equation:

$$\text{ElectionListSize} \rightarrow \text{Voters} + \text{Administrators Rate} + \text{Delay} \dots \dots \dots (4.1)$$

In order to simulate it by dizzy, the results are reasonable and better than REVS as the Figure 4.3 indicated. Noted that the command language parser is chosen as the default parser since the code is written in the CMDL that dizzy can understand it. After the parser is chosen, the simulate option is selected from tool menu and a dizzy simulator starting as indicated in Figure 4.2.

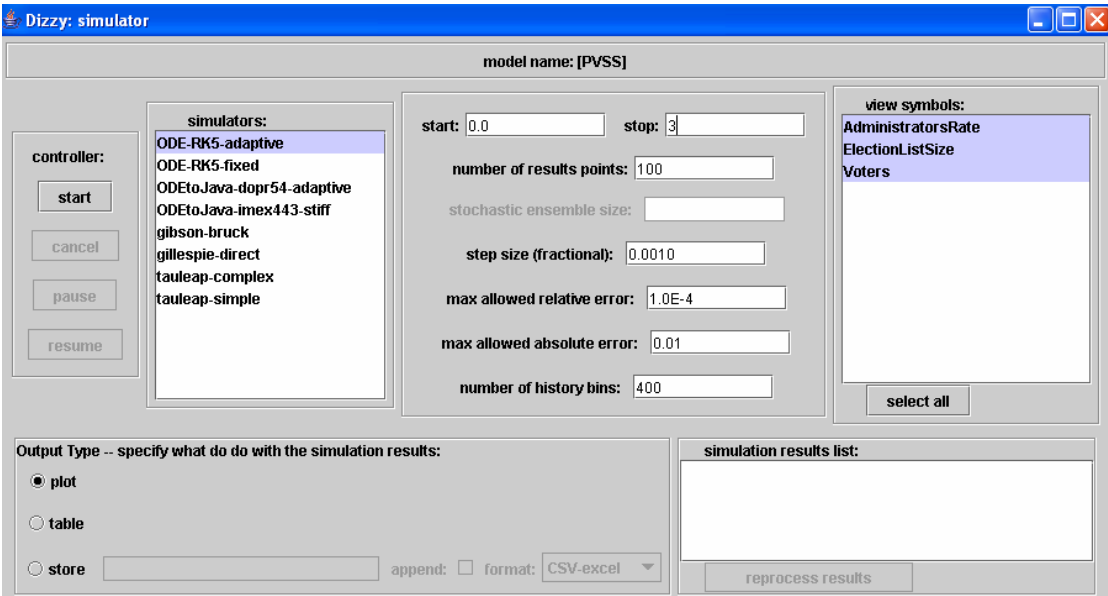


Figure 4.2: Dizzy simulator/PVSS

After choosing the options (view symbol→select all, stop→3, simulator type and starting), the relationship between voters, electionlistsize and administrators rate can be represented as Figure 4.3 shows.

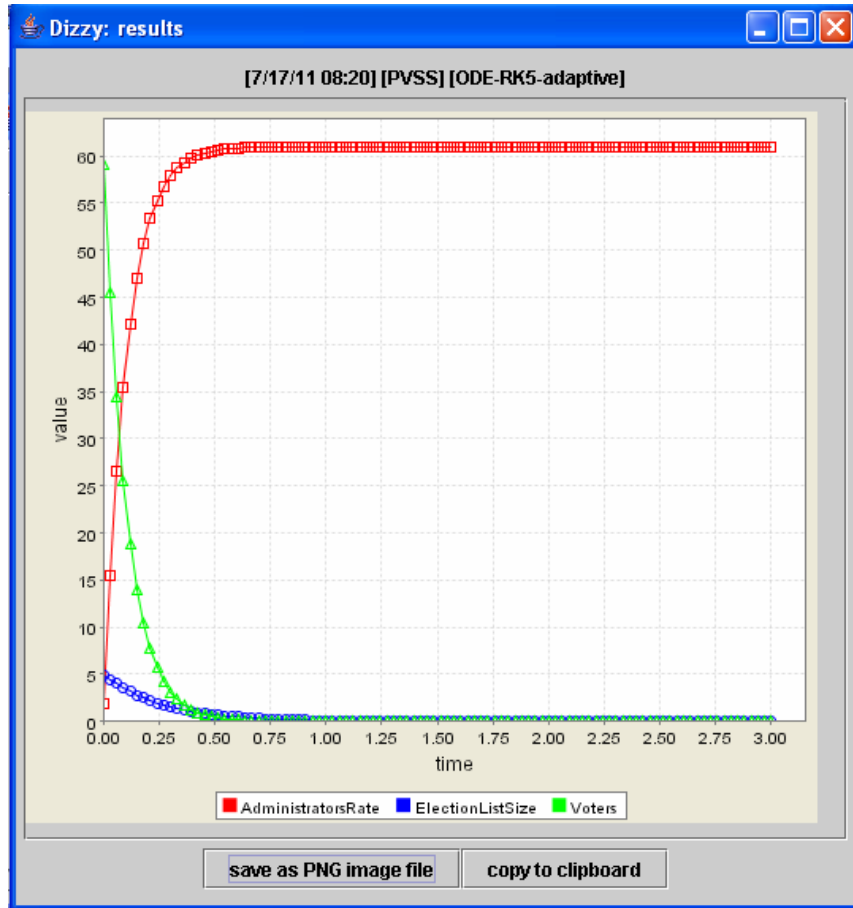


Figure 4.3: Dizzy Results/PVSS

4.2.1.1 Discussion

Under the assumption: The maximum number of administrators that will participate in ballot signing is n and at least $\geq n/2$, the maximum number of voters is the authenticated and registered only and the voter vote (electionlistsize) is fixed. Depending on the reaction equation (4.1), the researcher can conclude that at the beginning of the E voting process and depending on the proposed scheme dizzy results, the researcher notice that

- (1) At the rush hour the voters numbers are the maximum and as the time pass the number of voters participate will be decreased (and this is reasonable and make sense). noted that the proposed scheme prevent double voting as illustrated before (see Chapter Three).
- (2) The administrators' rate is the number of administrator involved in ballot signing (blind signature) must at least $\geq n/2$ and $\leq n$. As the number of administrators decreased (some may get defective or faulty), the time they need to reconstruct the

sharing depend on the PVSS will be less and consequently the delay to transfer the signing ballot will be less too.

At the time the E voting process is nearly complete, and depending on the above reaction equation, the proposed scheme will reach a steady state.

If the obtained results (presented in Table 4.8) compared with the REVS blind signature protocol results (Joaquim, et.al (2002)). The researcher can conclude that the stability of the proposed scheme in voting stage, specifically using PVSS to overcome the collude administrators problem, is higher than REVS.

Table 4.8: Proposed scheme simulation results

Dizzy: results			
[7/17/11 09:19] [PVSS] [ODE-RK5-adaptive]			
time	AdministratorsRate	ElectionListSize	Voters
0.00	1.8933	4.9701	59.107
0.030303	15.554	4.4741	45.446
0.060606	26.577	4.0036	34.423
0.090909	35.498	3.5509	25.502
0.12121	42.107	3.1493	18.893
0.15152	47.003	2.7932	13.997
0.18182	50.631	2.4773	10.369
0.21212	53.318	2.1972	7.6818
0.24242	55.309	1.9488	5.6910
0.27273	56.784	1.7284	4.2161
0.30303	57.877	1.5329	3.1234
0.33333	58.686	1.3596	2.3140
0.36364	59.286	1.2059	1.7143
0.39394	59.730	1.0695	1.2700
0.42424	60.059	0.94856	0.94086
0.45455	60.303	0.84130	0.69703
0.48485	60.484	0.74616	0.51638
0.51515	60.617	0.66179	0.38256
0.54545	60.717	0.58695	0.28341
0.57576	60.790	0.52058	0.20996
0.60606	60.844	0.46171	0.15555
0.63636	60.885	0.40950	0.11524
0.66667	60.915	0.36320	0.085371
0.69697	60.937	0.32213	0.063246
0.72727	60.953	0.28570	0.046855

Dizzy: results			
[7/17/11 09:19] [PVSS] [ODE-RK5-adaptive]			
time	AdministratorsRate	ElectionListSize	Voters
0.75758	60.965	0.25339	0.034712
0.78788	60.974	0.22474	0.025716
0.81818	60.981	0.19933	0.019051
0.84848	60.986	0.17679	0.014114
0.87879	60.990	0.15680	0.010456
0.90909	60.992	0.13907	0.0077463
0.93939	60.994	0.12334	0.0057387
0.96970	60.996	0.10939	0.0042515
1.0000	60.997	0.097023	0.0031497
1.0303	60.998	0.086052	0.0023334
1.0606	60.999	0.067691	0.0012807
1.0909	60.999	0.060036	0.00094876
1.1212	60.999	0.053247	0.00070287
1.1515	60.999	0.047226	0.00052072
1.1818	61.000	0.041886	0.00038577
1.2121	61.000	0.037150	0.00028579
1.2424	61.000	0.032949	0.00021172
1.2727	61.000	0.029223	0.00015685
1.3030	61.000	0.025918	0.00011620
1.3333	61.000	0.022988	8.6087E-5
1.3636	61.000	0.020388	6.3777E-5
1.3939	61.000	0.018083	4.7248E-5
1.4242	61.000	0.016038	3.5003E-5
1.4545	61.000	0.014224	2.5932E-5
1.4848	61.000	0.012616	1.9211E-5

Dizzy: results			
[7/17/11 09:19] [PVSS] [ODE-RK5-adaptive]			
time	AdministratorsRate	ElectionListSize	Voters
1.5152	61.000	0.011189	1.4232E-5
1.5455	61.000	0.0099240	1.0544E-5
1.5758	61.000	0.0088018	7.8113E-6
1.6061	61.000	0.0078065	5.7869E-6
1.6364	61.000	0.0069237	4.2871E-6
1.6667	61.000	0.0061408	3.1761E-6
1.6970	61.000	0.0054464	2.3530E-6
1.7273	61.000	0.0048305	1.7432E-6
1.7576	61.000	0.0042843	1.2914E-6
1.7879	61.000	0.0037998	9.5671E-7
1.8182	61.000	0.0033701	7.0877E-7
1.8485	61.000	0.0029891	5.2508E-7
1.8788	61.000	0.0026511	3.8900E-7
1.9091	61.000	0.0023513	2.8819E-7
1.9394	61.000	0.0020854	2.1350E-7
1.9697	61.000	0.0018496	1.5817E-7
2.0000	61.000	0.0016404	1.1718E-7
2.0303	61.000	0.0014549	8.6809E-8
2.0606	61.000	0.0012904	6.4312E-8
2.0909	61.000	0.0011445	4.7644E-8
2.1212	61.000	0.0010151	3.5297E-8
2.1515	61.000	0.00090029	2.6149E-8
2.1818	61.000	0.00079848	1.9372E-8
2.2121	61.000	0.00070819	1.4352E-8
2.2424	61.000	0.00062811	1.0632E-8

Dizzy: results			
[7/17/11 09:19] [PVSS] [ODE-RK5-adaptive]			
time	AdministratorsRate	ElectionListSize	Voters
2.2727	61.000	0.00055708	7.8768E-9
2.3030	61.000	0.00049409	5.8354E-9
2.3333	61.000	0.00043822	4.3231E-9
2.3636	61.000	0.00038866	3.2027E-9
2.3939	61.000	0.00034471	2.3727E-9
2.4242	61.000	0.00030573	1.7578E-9
2.4545	61.000	0.00027116	1.3022E-9
2.4848	61.000	0.00024050	9.6474E-10
2.5152	61.000	0.00021330	7.1472E-10
2.5455	61.000	0.00018918	5.2949E-10
2.5758	61.000	0.00016779	3.9226E-10
2.6061	61.000	0.00014882	2.9060E-10
2.6364	61.000	0.00013199	2.1529E-10
2.6667	61.000	0.00011706	1.5950E-10
2.6970	61.000	0.00010383	1.1816E-10
2.7273	61.000	9.2086E-5	8.7537E-11
2.7576	61.000	8.1673E-5	6.4851E-11
2.7879	61.000	7.2437E-5	4.8044E-11
2.8182	61.000	6.4246E-5	3.5593E-11
2.8485	61.000	5.6981E-5	2.6369E-11
2.8788	61.000	5.0538E-5	1.9535E-11
2.9091	61.000	4.4823E-5	1.4472E-11
2.9394	61.000	3.9754E-5	1.0721E-11
2.9697	61.000	3.5259E-5	7.9429E-12
3.0000	61.000	3.1272E-5	5.8844E-12

4.2.2 Counting stage –Trapdoor commitment scheme.

As the voter get his/her ballot signed by administrators and preventing collision problem. The voter is now ready to commit to the ballot using a trap door commitment scheme and sending it anonymously via anonymizers to the counter for counting purpose (see Chapter Three :3.3 Counting stage). In order to use dizzy simulator the researcher presents the communicating entities as a reaction equation:

Voters+PVIDlist → Counters+Delay..... (4.2)

The analysis will represent the relationship between the communicating entities in the equation (see Figures 4.4). The result will be reasonable also and make sense depend on the proposed scheme.

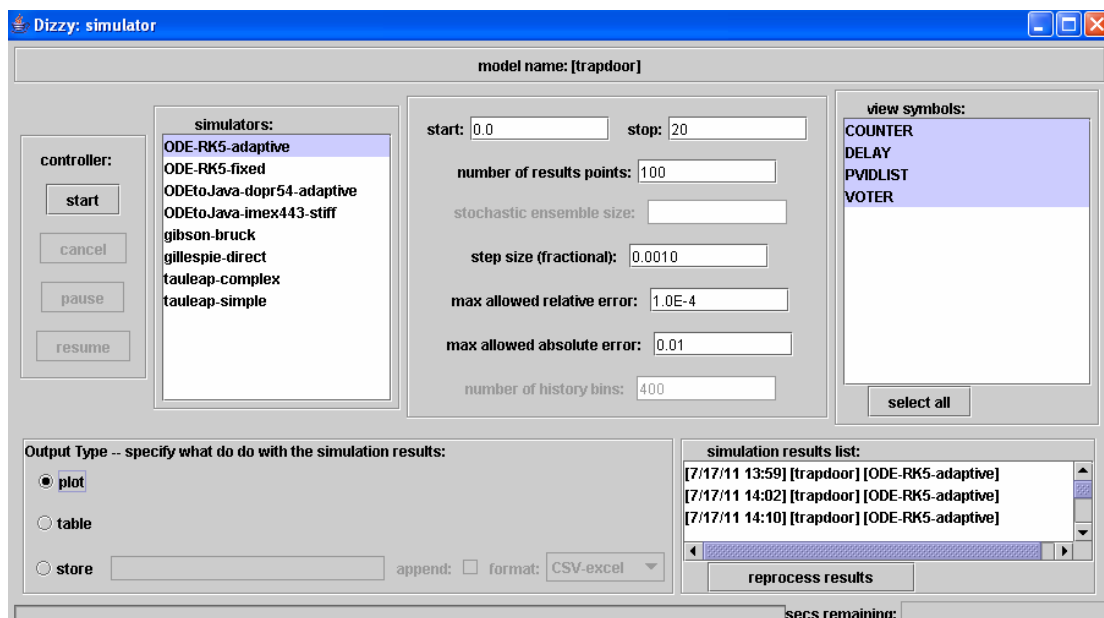


Figure 4.4: Dizzy simulator /Trapdoor commitment scheme

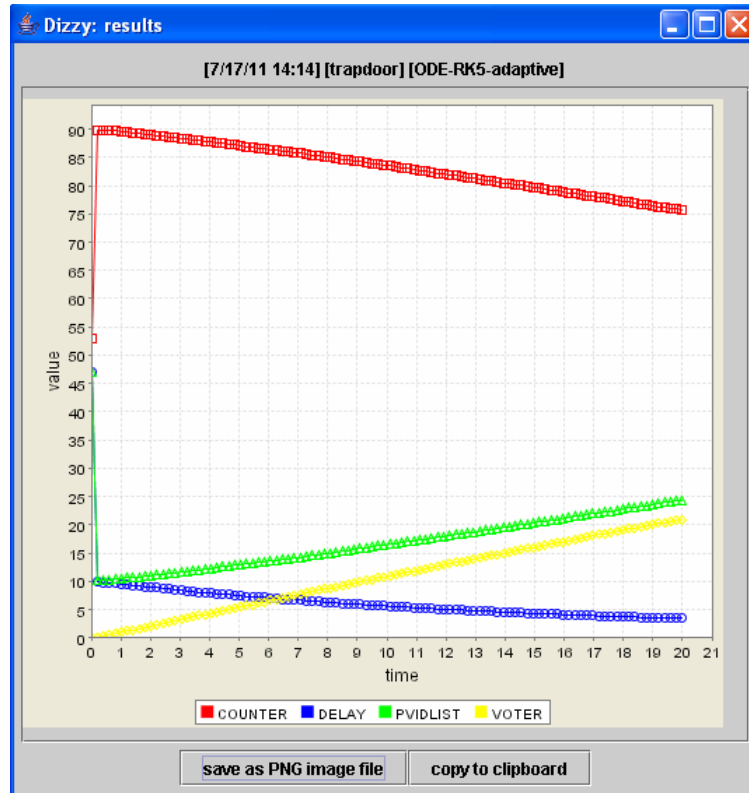


Figure 4.5: Dizzy Results / Trapdoor commitment scheme

4.2.2.1 Discussion

Under the assumption: The maximum number of voters is the authenticated and registered only, PVID list accompanied with each voter for authentication purpose, and counters number may decrease as the time pass as one may get defective or faulty server. Depending on the reaction equation (4.2), the researcher can conclude that at the beginning of the E voting process and depending on the proposed scheme dizzy results, the researcher notice that:

(1) The voters number at the beginning were less and then increased the same behaviour will be for the PVID as each voter accompanied authenticated PVID. Noted that by using a trap door commitment scheme a double voting is prevented as illustrated before (see Chapter Three).

(2) The results show that as the number of voters increased and thus PVID increased, the delay time in which voters committed to their ballot in addition to the time comparison for PVID match between the commissioner and counter database will get stable due to the reference of the trapdoor commitment scheme even if the counters number decreased as the time pass by (see Figure 4.5).

Chapter 5

Conclusion and Future work

5.1 Conclusion

In this thesis, the researcher have proposed a secure E voting scheme depending on the last two E voting protocols based on the blind signature, Evox-MA and REVS; it encompasses three distinct phases, which of registration phase, voting phase and counting phase. Each phase applies some cryptographic technique, schemes or modified protocol to enhance some security aspects as a Kerberos authentication protocol, PVID scheme, responder certificate validation. The theoretical proof and simulation results show that the scheme satisfies all E voting security requirements. And by applying a PVSS the researcher get a more stable result than REVS blind signature protocol which suggests using a different password for each administrator. Also the proposed scheme adds more security enhancements

- (1) By applying more than one scheme, the Kerberos authentication protocol (it has been modified by adding a new entity (responder) that derived from the OSCP-KIS protocol to verify voter certificate validity), PVID scheme and the converted Ferguson E cash protocol the researcher guarantee that only authorized voters vote. Therefore, limit the DoS attack against attackers, so the counter buffer will never be filled with a garbage votes
- (2) Detecting the double voting issued by the voters, by applying more than one mechanism (the converted Ferguson E cash protocol to operated under E voting, trapdoor commitment scheme, and modified PVID scheme
- (3) Allow a valid vote to be repeated if fault tolerance occurred by applying a trapdoor commitment scheme.

5.2 Future work

As a recommendation for the proposed secure scheme, the researcher suggests to extend the scheme to the image processing field by applying the visual cryptography techniques for voter authentication purpose. Also the researcher encourages deploying the scheme under mobile network and comparing the result between such infrastructure and Kerberos authenticated framework. Furthermore, the researcher encourages combining Mixing net scheme with the blind signature scheme used in the proposed scheme, to add more values for the secure proposed scheme and to apply some synchronization algorithm for simultaneous voting.

References:

Abe M. (1998): "Universally verifiable mix-net with verification work independent of the number of mix servers", in advances in Cryptology –EUROCRYPT'98, *SpringerVerlag*, Volume 1403, pp.437-447.

Acquisti A. (2004): "Receipt free homomorphic elections and write-in voter verified ballots", *ISRI Technical report CMU-ISRI-04-116*, Carnegie Mellon University, PA.

Aggelos Kiayias Michael Korman, and David Walluck: "An Internet Voting System Supporting User Privacy ", IEEE, ISSN: 1063-9527, 0-7695-2716-7, pp 165-174, 2006.

Baek J., Zhen Y.(2004): "Identity-Based Threshold Decryption", Cryptology ePrint Archive, Report 2003/164, available at <http://eprint.iacr.org/2003/164>, last access 14th July 2011.

Benaloh J.C (1987): "A verifiable secret-ballot elections, PhD thesis (published), New Haven, Yale University, and Institute of information Technology, USA.

Benaloh J, and Franklin MK. (2001): "Efficient generation of shared RSA Keys" ,*Journal of the ACM(JACM)*, vol. 48-4, pp. 702-722, New York, USA.

Benaloh J, and Tuinstra D. (1994): "Recipet free secret ballot elections", in proceedings of the 26th ACM Symposium on the theory of computing, pp.554-553, *IEEE presses*.

Boyd, C. (1989): "A new multiple key cipher and an improved voting scheme". Advances in Cryptology EUROCRYPT '89, *Springer-Verlag*.

Bresson E ,Catalano D, and Pointcheval (2003) : “A simple public key cryptosystem with a double trapdoor decryption mechanism and its applications”. In Laih CS, ed. *Aciacrypt 2003*. LNCS 2894, pp.37-54 , *Berlin: Springer-Verlag*.

Camenisch J., and Lysyanskaya A. (2005): "A formal treatment of onion routing", In Advances in cryptology- CRYPTO'05, *SpringerVerlag*, pp.169-187.

Cavadini & Lorenzo ,D.C, (2007): "Case studies on E Voting", *university of Fribourg ,seminar work on E-government* supervised by Professor Dr. Andreas Meier.

Cetinkaya O. & Doganaksoy (2007): "Pseudo-Voter Identity (PVID) Scheme for E voting Protocols" *METU*, Ankara, Turkey, Institute of Applied Mathematics and computer science.

Cetinkaya O, and Koc M.L (2009): "Practical Aspects of DynaVote E Voting Protocol.", *Electronic Journal of E Government*, Volume 7 Issue 4, (pp327 - 338), available online at www.ejeg.com.

Chaum D. (1981): "Untraceable electronic mail, return addresses, and digital pseudonyms", *Communications of the ACM journal*, vol 24, pp 84-90.

Chaum D. (1983): "Blind signatures for untraceable payments", In Proceedings of Crypto 82, page 199 – 203, *Plenum Press*, New York.

Chaum, D. (1988): "Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA", advances in cryptology EUROCRYPT *Springer-Verlag*, Amsterdam.

Chaum, D. L. (1992): "Achieving Electronic Privacy", *Scientific American*, pp. 96-101, USA.

Chaum D. (2004): "Secret ballot receipt :True voter verifiable elections" *IEEE press Security &privacy* ,Vol.2-1 ,pp 38-47 .

Chaum D. & Pedersen T.P (1993): "Wallet databases with observers", In Advances in Cryptology—CRYPTO '92, volume 740 of Lecture Notes in Computer Science, pages 89–105, *Springer-Verlag*.

Chen X, Wub O, Zhang F, Tian H, Wei B, Lee B, Lee H, Kim K (2011): "New receipt-free voting scheme using double-trapdoor commitment", *Information Science Magazine*, pp.1493-1502.

Cohen J. and Fischer M. (1985): "A robust and verifiable cryptographically secure election scheme", in Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science (FOCS), pp 372 – 382, *IEEE Press*.

Cohen J. and Yung M. (1986): "Distributing the power of government to enhance the privacy of voters", in Proceedings of 5th ACM Symposium on Principles of Distributed Computing (PODC), pp 52-62, *IEEE Press*.

Cramer R., Gennaro R., Schoenmakers B., and Yung M. (1996): "Multi-Authority Secret-Ballot Elections with Linear Works", *Springer-Verlag*, Vol. 1070 of Lecture Notes in Computer Science, pp. 72-83.

Cramer R., Gennaro R., and Shoenmakers B. (1997): "A secure and optimally efficient multi-authority election scheme", in Advances in Cryptology – Eurocrypt 97, *SpringerVerlag*, pp. 103 -118, LNCS.

Cranor LR, and. Cytron R.K (1997): "Design and Implementation of a Practical Security-Conscious Electronic Pollind System", in proceeding of Thirtieth Hawaii International, *Wilea HI*, USA , vol.3 ,pp 561-570.

Cranor & Cytron, L.R(1997): "Sensus: a security-conscious electronic polling system for the Internet. *Hawaii International Conference on System Sciences*, Wailea, Hawaii.

Davenport B., Newberger A, and Woodar J. (1996): "Creating a Secure Digital Voting Protocol for Campus Elections", *Princeton University*, Department of computer engineering and computer Science, UK.

Desmedt Y.(1993): "Threshold Cryptosystems", *Advances in Cryptology-ASIACRYPT92*, Old Coast, Queensland

DuRette B.W, (1999): "*Multiple Administrators for Electronic Voting*", Msc. Thesis (published), Massachusetts Institute of Technology MIT, Cambridge, USA.

"E-vote: Election markup language 5.0 approved as OSAIS standard" (2008), *New Report government technology magazine*, (On-Line), available:<http://www.govtech.com/e-government/E-Vote-Election-Markup-Language-50-Approved.html>. Last access on 21 March 2011

Fujioka A., Okamoto T., & Ohta K. (1992): "A practical secret voting scheme for large scale elections", proceedings on the theory and application of cryptographic techniques, pp.244-251, *Springer Verlag*, Australia.

Furukawa J. and Sako K. (2001): "An efficient scheme for proving a shuffle", in advances in cryptology-CRYPTO'01, *SpringerVerlag*, pp.368-387, Berlin, Germany.

Goldschalg D., Reed M., and Syverson P. (1999): "Onion routing for anonymous and private communications", *Communications of the ACM*, vol. 42-2, pp.39-41.

Goldwasser S., Micali S., and Rackoff C. (1985): "The knowledge complexity of interactive proofs", *In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291-305.

Golle P., Jakobsson M., Jules A., and Syverson P. (2004): "Universal re-encryption for mix nets", In Proceedings of CT-RSA'04, pp.163-178, *IEEE press*.

Harris R. (1999): " Voting-By-Mail: A Look at Modernizing the Electoral System, vol. 6. no. 3, *California state library foundation*, California, USA.

Hirschberg M. (1997): "Secure electronic voting over the World Wide Web", Bsc. Thesis (published), Massachusetts Institute of Technology MIT, Cambridge, USA.

Hirt M. and sako K. (2000): "Efficient recipient free voting based on homomorphic encryption", *EUROCRYPT*, pp.539-556, Bruges, Belgium.

Horster P., Michels M., and Petersen H. (1995): " Blind Multi signature Schemes and their Relevance to Electronic Voting", in the proceeding of 11th Annual Computer Security Applications Conference, pp. 149-155, *IEEE Press*, Germany.

Jakobsson M., and Juels A. (2001): "An optimally robust hybrid mix network", In proceedings of the 20th Annual ACM Symposium of Principles of Distributed Computing (PODC 01), pp.284-295, *ACM Press*.

Jakobsson M., Jules A. and Rivest R. L (2002): "Making mix nets robust for electronic voting by randomized partial checking", in Proceedings of the 11th USENIX Security Symposium, pp.339-353, *IEEE press*.

Joaquim R., Zúquete A., Ferreira P. (2002): "REVS –a roubst electronic voting system", *Instituto Superior Técnico (Technical Univ. of Lisbon) / INESC ID* , Lisboa, Portugal.

Juang W. S., Lei C. L, and Yu P. L. (1998): "A verifiable multi-authorities secret election allowing abstaining from voting", *International Computer Symposium*, Tainan, Taiwan.

Kalaichelvi. V, Chandrasekaran R. (2011): "Secured single transaction E voting protocol design and implementation", *European Journal of Scientific Research*, Vol.51 No.2 (2011), pp.276-284.

Kelsey J., Schneier B., Wagner D. & Hall C. (1997):" Cryptanalytic Attacks on Pseudorandom Number Generators", *LATEX macro package with LLNCS style*

Kohno T., Stubblefield A., Rubin A., Wallach D. (2004):" Analysis of an Electronic Voting System", *IEEE*.

Lagnana A. et. al (2004): "Computational science and its application-ICCSA 2004",international conference proceedings, Assisi, Italy, *Springer-Verlag*.

Lee X. (2010): "Security analysis on an elementary e-Voting System", *IJCSNS International Journal of Computer Science and Network Security*, VOL.10 No.10, New York, USA.

Libert B. and Quisquater J.(2003): "Efficient Revocation and Threshold Pairing Based Cryptosystems", *Symposium on Principles of Distributed Computing PODC*, pp. 163-171.

Meng B, Li Z. and Qin J (2010): "A Receipt-free coercion-resistant remote internet voting protocol without physical assumptions through deniable encryption and trapdoor commitment scheme", *JOURNAL OF SOFTWARE*, VOL. 5, NO.9, China.

Menezes A.(1993): "Elliptic Curve Public Key Cryptosystems", *Kluwer Academic Publishers*

Menezes A.J, Van Oorschot P.C, and Vanstone A. S. (1997): "Hand- book of Applied Cryptography", *CRC Press*.

Migeon J.(2008): " The MIT Kerberos Administrator's How-to Guide Protocol, Installation and Single Sign On", *the MIT Kerberos Consortium*, version 23,pp.1-62.

Mote Jr, (2001): "Report of the National Workshop on Internet Voting: Issues and Research Agenda", *Internet Policy Institute*, Maryland.

Naznin F., Dey T., Bhuiyan I, Saidur R. (2007): "An efficient implementation of electronic election system", *IEEE Press*, Bangladesh.

Neff C.A (2001): "A verifiable secret shuffle and its application to E voting", in proceedings of the 8th ACM conference on computer and communications security, pp 116-125, *IEEE press*.

Nurmi H., Salomaa A., and Santeau L. (1991): "Secret ballot elections in computer networks" ,*Computers and Security*, pp.553-560.

Okamoto T. (1997): "Receipt free electronic voting schemes for large scale elections", in Proceedings of Workshop on Security Protocols, vol 97, pp 25-35 *Springer Verlag*, LNCS.

Park C., Itoh K., and K. Kurosawa (1993): "Efficient anonymous channels and all/nothing election scheme", In Advances in cryptology- EUROCRYPT'93, *SpringerVerlag* , pp.248-259, Lofthus, Norway

Pfitzmann B. (1994): "Breaking efficient anonymous channel", In Advances in cryptology- EUROCRYPT'94, *SpringerVerlag*, pp.332-340, Italy.

Radwin, M. J. (1995): "An untraceable, universally verifiable voting scheme". Seminar in Cryptology under Professor Phil Klein supervision.

Ramsey S. (2006):" Dizzy User Manual", *CompBio Group, Institute for Systems Biology*, Dizzy Version: 1.11.4.

Request for comment 3147 (2001): "US Secure Hash Algorithm 1 (SHA1)", Network working group, Cisco system

Request for comment 4868 (2007): "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 ", Network working group, Cisco system

“Report of the National Workshop on Internet Voting: Issues and Research Agendas,” Internet Policy Institute, Sponsored by the National Science Foundation, Conducted in cooperation with the University of Maryland and hosted by the Freedom Forum, March 2001.

Rivest R. (1994): "The MD5 Message Digest Algorithm", RFC 1320, *MIT and RSA Data Security*.

Rivest R. (2000): "Electronic Voting", Laboratory for Computer Science *Massachusetts Institute of Technology (MIT)*, Cambridge

Rivest R., Shamir A., and Adelman L.M (1977): "A method for obtaining digital signatures and public-key cryptosystems, *MIT LCS Technical Report MIT/LCS/TM*.

Sako K., and Kilian J. (1994): "Secure voting using partially compatibles homomorphism", in advances in Cryptology Eurocrypt 94, *SpringerVerlag*, pp.411-424, Santa Barbara, USA.

Sako K. and Kilian J. (1995): "Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth", in Advances in cryptology-EUROCRYPT'95, *SpringerVerlag*, pp.393-403, Malo, France.

Schryen G. (2004): "Security aspects of internet voting", Proceedings of the 37th Hawaii International Conference on System Sciences, *IEEE press*.

Scarfone K., Hoffman P. (2009): "Guidelines on Firewalls and Firewall Policy", *NIST Special Publication*, pp.800-41, Revision 1, Germany.

Shamir A. (1979): "How to share a secret", 22(11):612– 613, *Communications of the ACM*.

Stadler M. (1996): " Publicly Verifiable Secret Sharing", in Advances in Cryptology - EUROCRYPT'96, volume 1070 of Lecture Notes in Computer Science, pp. 190-199, *Springer Verlag*.

Subariah I, Maznah K, Mazleena S., Shah Rizan A.Z (2001): "Secure E voting with blind signature", *Faculty of Computer Science & Information Technology University Technology of Malaysia*, Malaysia

Subariah I, Maznah K, Mazleena S., Shah Rizan A.Z (2003): "Electronic voting system: preliminary study", *Faculty of Computer Science & Information Technology University Technology of Malaysia*, Malaysia

Wei-Chi K., Sheng-De W. (1999): "A secure and practical electronic voting scheme", *Elsevier Science*, pp.286–279, Taiwan.

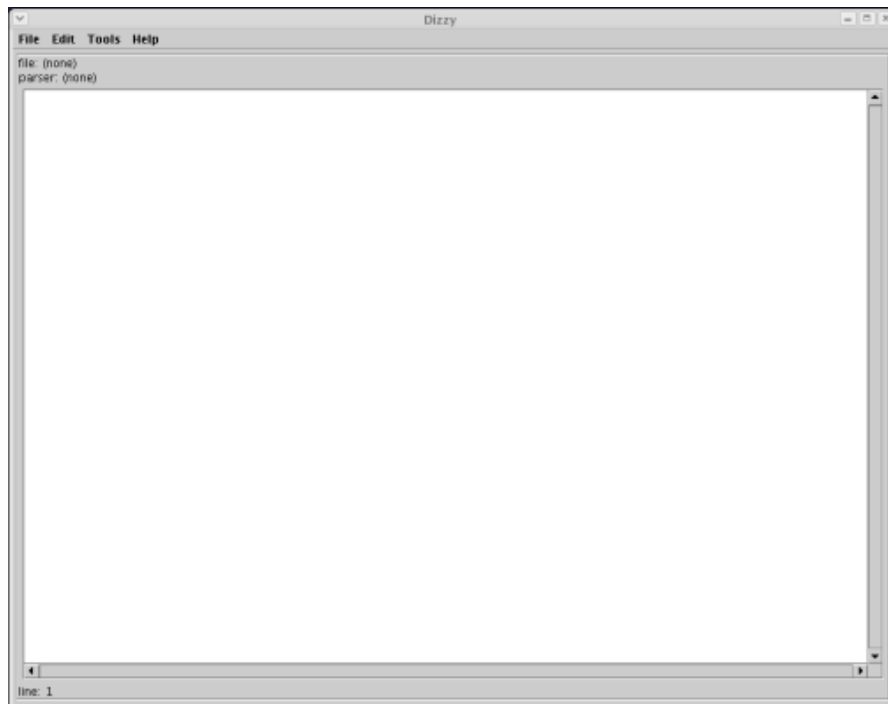
Wen X., Niu X., Liping J and Tian Y. (2009): "A weak blind signature scheme based on quantum cryptography", Volume 282, Pages 666-669, *IEEE*.

Appendices:

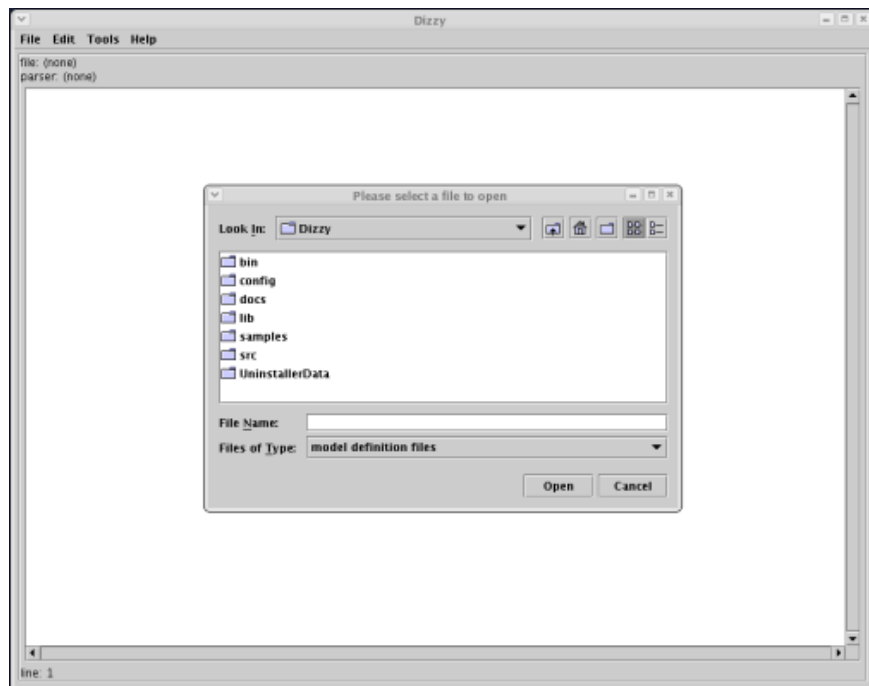
Appendix A:

Using Dizzy simulator Tutorial

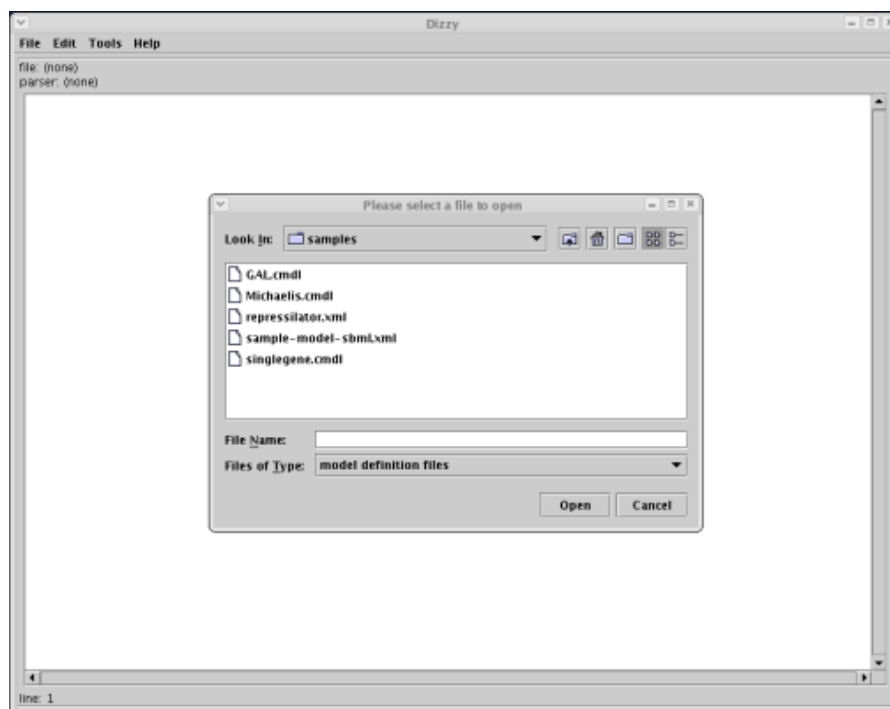
Dizzy is launched by executing the "Dizzy" executable that was installed as a symbolic link by the installation program. The default location of this symbolic link depends on your operating system. If you are installing on a Windows computer, the symbolic link is created in a new Program Group "Dizzy", which will show up in the "Start" menu. If you are installing on a Linux computer, the symbolic link is created in your home directory, by default. Note that the installation program permits you to override the default location for the symbolic link to be created, so the symbolic link may not be in the default location on your computer, if you selected a different location in the installation process. By double-clicking on the "Dizzy" symbolic link, the Dizzy program should start up. You should see an application window appear that looks like the following picture:



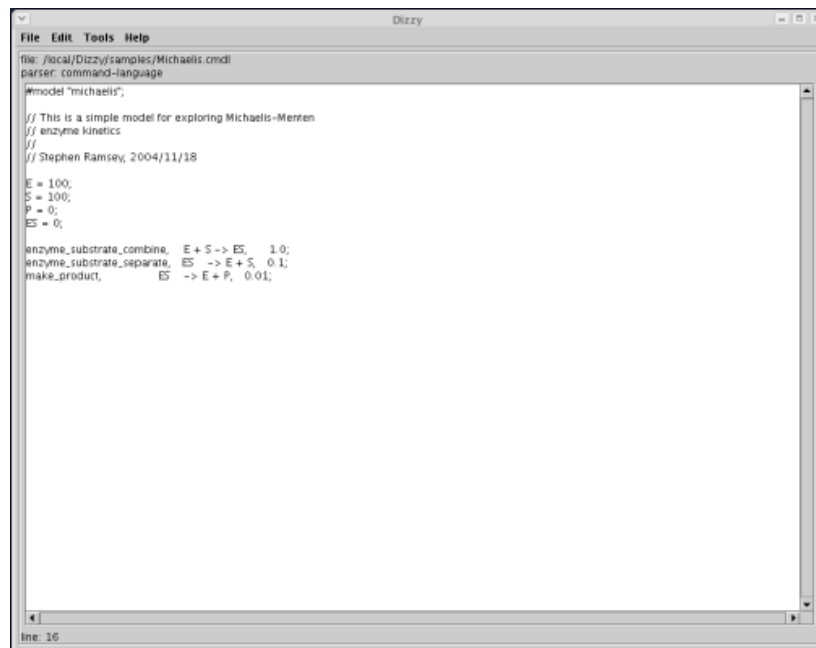
To load a model definition file into Dizzy, select the "Open..." item from the "File" menu. This will open a dialog box, as shown here:



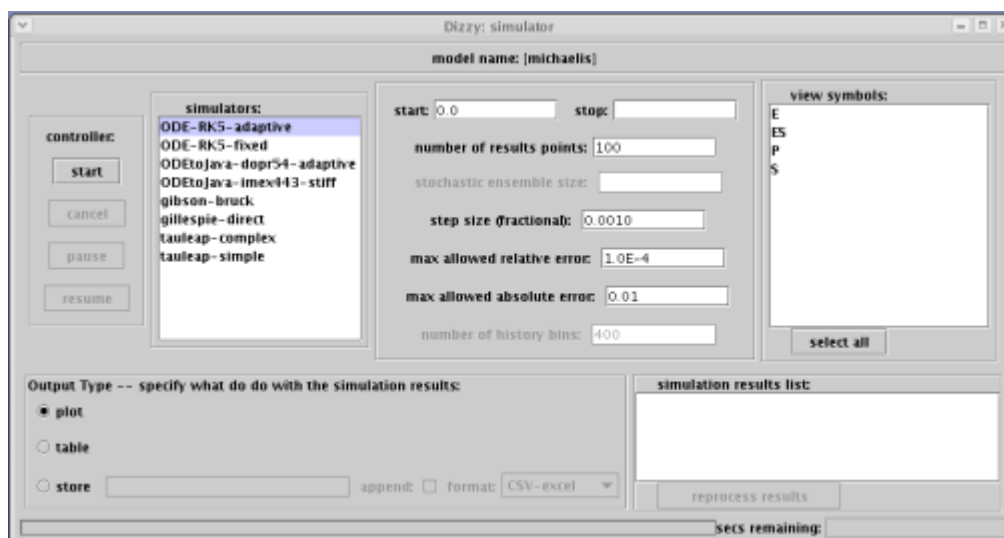
In the "Please select a file to open" dialog box, navigate to the directory in which you installed Dizzy. Then navigate into the "samples" subdirectory. The dialog box should look like this:



For starters, try selecting the "Michaelis.cmdl" file, by double-clicking on that file name in the dialog box. The Dizzy window should now look like this:



Note that the model description has appeared in the editor window. In this window, you can edit a model description, after which you may save your changes. You probably will not want to modify the Michaelis.cmdl model definition file just yet. Note that the file name appears after the "**file:**" label. There is also a label "**parser:**" label, whose function will be described later. Now, from the "Tools" menu, select "Simulate..." which essentially processes the model definition and loads the relevant information into the dizzy simulation engine. This should create a "Dizzy: simulator" dialog box, that looks like this:

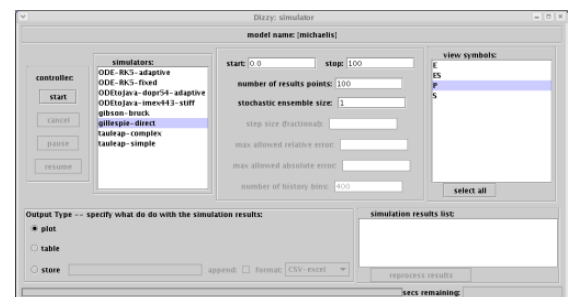


First, you will need to specify a "stop time" for the simulation. This is a floating-point number that you must type into the text box next to the "stop time:"

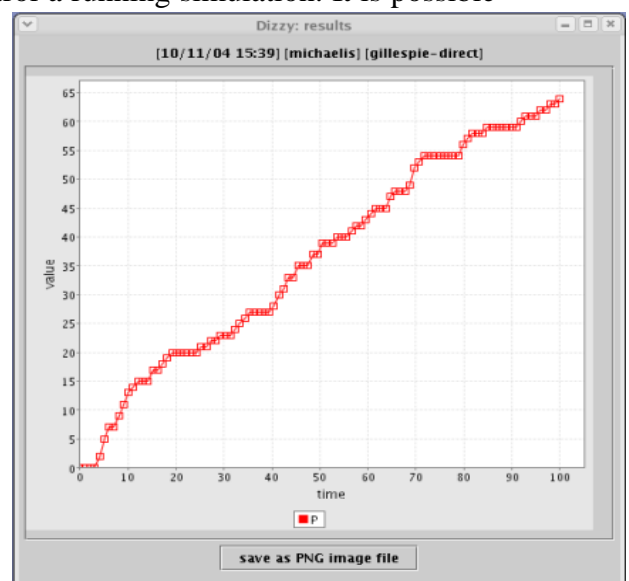
label in the "Dizzy: simulator" dialog box. Second, you will need to select one or more species whose populations are to be returned as time-series data resultant from the simulation. For the purposes of demonstration, select the "G3D_G80D" species in the list box under the "view species" label in the dialog box.

TIP: You can select two species that are not adjacent to one another in the list box of species, by holding down the "control" key, and (while holding down the key) clicking on a species name with the mouse.

Finally, you will need to specify the "output type" for the simulation. For demonstration purposes, click on the circular button next to the "plot" label on the dialog box. Go ahead and change the number of samples to 30 samples, by editing the "100" appearing in the text box next to "num samples". This controls the number of time points for which the result data will be graphed. At this point, the dialog box should look like this:

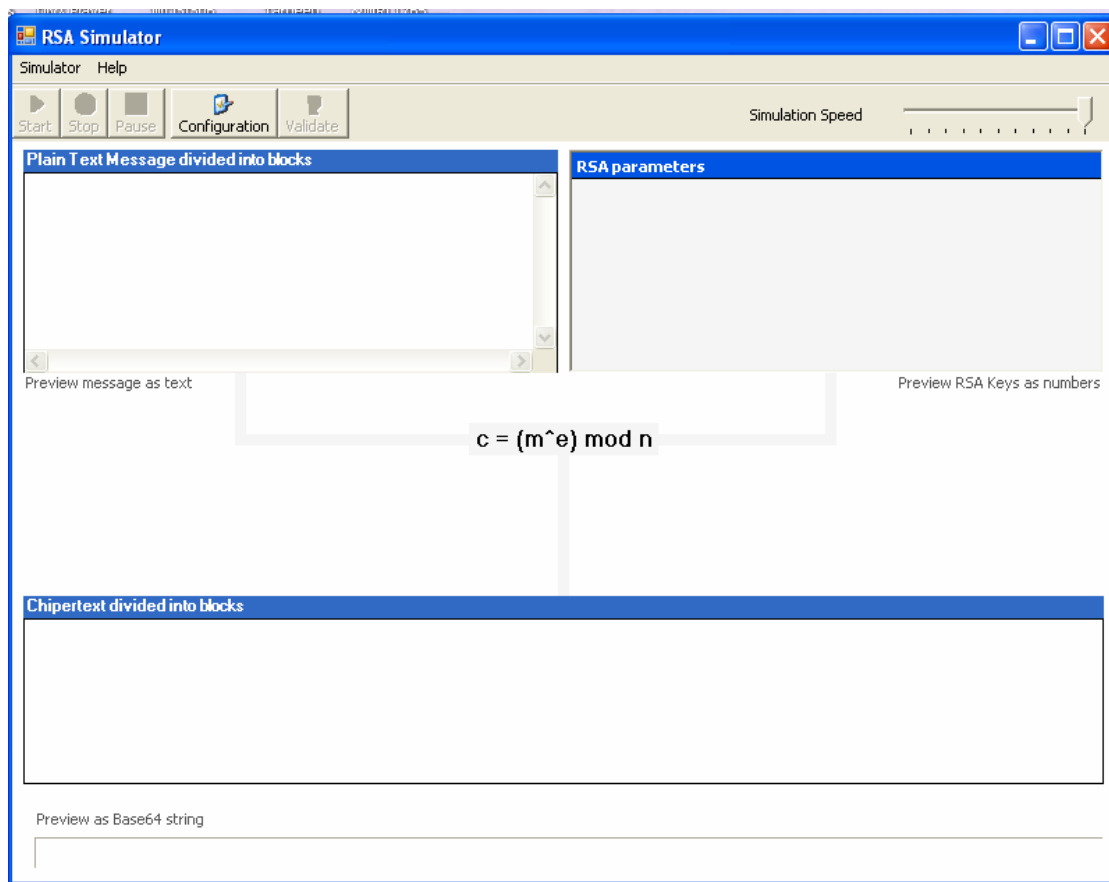


Now, let's run the simulation, by single-clicking on the "start" button in the "Dizzy: simulator" dialog box. After a moment, you should see a plot window appear that resembles the following image, For longer-running simulations, you can use the "cancel", "pause", and "resume" buttons to control a running simulation. It is possible to pause and resume a simulation using the "pause" and "resume" buttons. You may terminate a running simulation at any time using the "cancel" button. The "start" button is only used to initiate a simulation. Only one simulation may be running at a time, in the dizzy application.



Appendix B:

RSA simulator



Configuration

Examples

Plain text:
Format:
☐ Integer
☒ String

Message...

Keys

☒ Simulate RSA Algorithm with random keys
☐ Simulate RSA Algorithm with own keys

Key size
16
bits.

Public Exponent (e)
(random)

☒ Generate randomly!

p

q

n
 $n=p*q$

☒ Calculate!

phi
 $\phi=(p-1)*(q-1)$

☒ Calculate!

Public Exponent (e)
 $1 < e < \phi, \gcd(e; \phi) = 1$

☒ Calculate!

Private Exponent (d)
 $1 < d < \phi, ed = 1 \pmod{\phi}$

☒ Calculate!

OK

RSA Simulator

Simulator Help

Start Stop Pause Configuration Validate Simulation Speed

Plain Text Message divided into blocks

Block0: 01001101
Block1: 00000000
Block2: 01100101
Block3: 00000000
Block4: 01110011
Block5: 00000000
Block6: 01110011
Block7: 00000000
Block8: 01100001
Block9: 00000000

[Preview message as text](#)

RSA parameters

n: 0000101110000111
Public Exponent: 0000000010101101
Private Exponent: 0011110101111001

[Preview RSA Keys as numbers](#)

$c = (m^e) \bmod n$

Chiphertext divided into blocks

[Preview as Base64 string](#)

RSA Simulator

Simulator Help

Start Stop Pause Configuration Validate Simulation Speed

Plain Text Message divided into blocks

Block11: 00000000
Block12: 01100101
Block13: 00000000
Block14: 00101110
Block15: 00000000
Block16: 00101110
Block17: 00000000
Block18: 00101110
Block19: 00000000

[Preview message as text](#)

RSA parameters

n: 0000101110000111
Public Exponent: 0000000010101101
Private Exponent: 0011110101111001

[Preview RSA Keys as numbers](#)

$c = (m^e) \bmod n$

Chiphertext divided into blocks

Block11: 0000000000000000
Block12: 0001100001101001
Block13: 0000000000000000
Block14: 0000100100110010
Block15: 0000000000000000
Block16: 0000100100110010
Block17: 0000000000000000
Block18: 0000100100110010
Block19: 0000000000000000

[Preview as Base64 string](#)

Simulation complete!

OK

