AUDITABLE AND VERIFIABLE ELECTRONIC VOTING WITH HOMOMORPHIC RSA TALLYING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

OKAN YÜCEL

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JULY 2010

Approval of the Graduate School of Informatics

____

Prof. Dr. Nazife BAYKAL

Director

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

____

Assist. Prof. Dr. Tuğba Taşkaya Temizel

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as thesis for the degree of Doctor of Philosophy.

____

Prof. Dr. Nazife BAYKAL

Supervisor

Examining Committee Members:

Assist. Prof. Dr. Altan KOÇYİĞİT	(METU, II)	______________________
Prof. Dr. Nazife BAYKAL	(METU, II)	______________________
Assoc. Prof. Dr. Kemal BIÇAKCI	(TOBB)	______________________
Assist. Prof. Dr. P. Erhan EREN	(METU, II)	______________________
Assoc. Prof. Dr. Murat ERTEN	(INNOVA)	______________________

ABSTRACT

AUDITABLE AND VERIFIABLE ELECTRONIC VOTING WITH HOMOMORPHIC RSA TALLYING

Yücel, Okan

Ph.D., Department of Information Systems

Supervisor: Prof. Dr. Nazife Baykal

July 2010, 126 pages

In this work, we investigate the general structure and the concepts behind the contemporary electronic voting schemes, with special emphasis on voter verifiable preferential voting, homomorphic tallying and voter privacy. We firstly propose a modification in the Single Transferable Voting (STV) method to be applied to large scale elections with electoral barriers. Our proposal prevents the loss of votes and distributes them securely to the second or higher choices of their voters. This method is most suitably used in e-voting with the voter verifiable “Prêt à Voter: All-In-One” scheme that utilizes mix-networks for anonymity.

We present a case study considering 2007 Turkish Parliamentary Elections to demonstrate the effect of preferential voting on the election systems that have electoral barriers. After the mathematical formulation of the election procedure, we calculate the wasted votes in 2007 elections and present simulation results for 69 election regions (that have no independent parliament members) by using a combination of “modified STV and d’Hondt” methods, according to four different, politically unbiased scenarios on the distribution of secondary vote choices.

Additionally, we modify the “Prêt à Voter: All-In-One” scheme by proposing three security enhancing modifications in its ballot construction phase: 1) ballot serial number, 2) digital signature of the first clerk in the mix-net, 3) different random numbers for each row of the ballot.

Finally, we demonstrate the potential of multiplicative homomorphic algorithms like RSA for homomorphic tallying. The idea is based on the association of each candidate on the electronic ballot with a prime number, and unique prime factorization of the general vote product. We propose novel randomization methods for homomorphic RSA tallying, and discuss the performance and complexity of the scheme with such randomizations. Our suggestion for an auditable and verifiable e-voting scheme that employs homomorphic RSA tallying with proper randomization has advantages over El Gamal and Paillier tallying, such as having the least encryption complexity and strong anonymity resistant to unlimited computational power.

Keywords: anonymity, e-voting, homomorphic tallying, mix-nets, preferential voting, RSA randomization, single transferable voting, voter verifiability.

ÖZ

HOMOMORFİK RSA SAYIMLI, İZLENEBİLİR VE DENETLENEBİLİR ELEKTRONİK OYLAMA

Yücel, Okan

Doktora, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Prof. Dr. Nazife Baykal

Temmuz 2010, 126 sayfa

Bu çalışmada, çağdaş elektronik oylama sistemlerinin genel yapısı ve arka planındaki kavramlar, seçmen izlemeli ve tercihli oylamalarla, homomorfik sayımlar ve seçmen gizliliğine özel vurguyla incelenmiştir. Öncelikle, Tek Geçişli Oylama (TGO) yönteminin, seçim barajı içeren büyük ölçütlü seçimlerde kullanımına yarayan bir değişiklik önerilmiştir. Bu önerimiz, seçim barajı altında kalan partilerin oylarını, seçmenlerinin ikinci veya daha sonraki tercihlerine güvenli bir şekilde dağıtarak, oyların ziyan olmasını engelleme amaçlıdır. E-oylama için en uygunu, bu yöntemin, seçmen izlemeli ve anonimliği sağlamak için karıştırıcı ağlardan yararlanan “Prêt à Voter: All-In-One” ile birlikte kullanımıdır.

2007 Türk Parlamenter seçimleri için, tercihli oylamanın barajlı seçimlere etkisini gösteren bir örnek çalışma yapılmıştır. Seçim prosedürünün matematiksel formülasyonundan sonra, 2007 seçimlerinde bağımsız aday çıkarmamış olan 69 seçim bölgesinde boşa giden oylar hesaplanmış; ve seçmenlerin ikincil tercihleri üzerinde dört tarafsız senaryoya göre, “değiştirilmiş TGO ve d’Hondt” yöntemlerini birlikte kullanan simülasyon sonuçları sunulmuştur.

Çalışmamızda ayrıca, “Prêt à Voter: All-In-One” yönteminin oy pusulası hazırlama fazı için üç güvenlik arttırıcı değişiklik önerilmiştir: 1) pusula seri numarası 2) karıştırıcı ağdaki ilk görevlinin sayısal imzası, 3) oy pusulasının her satırı için ayrı bir rasgele sayı üretilmesi.

Son olarak, RSA gibi çarpmaya göre homomorfik algoritmaların homomorfik sayım açısından potansiyeli gösterilmiştir. Anafikir, elektronik oy pusulasında her adayın bir asal sayıyla ifade edilmesine ve genel oy çarpımının yalnız tek bir şekilde asal çarpanlarına ayrılabilmesine dayanmaktadır. Homomorfik RSA sayımı için farklı rassallaştırma (rasgeleleştirme) yöntemleri önerilmiş; bu durumdaki başarım ve karmaşıklık tartışılmıştır. Önerdiğimiz rassallaştırılmış homomorfik RSA sayımlı, izlenebilir ve denetlenebilir e-oylama yönteminin, homomorfik El Gamal ya da Paillier sayımıyla karşılaştırıldığında, en az şifreleme karmaşıklığı gerektirme ve sınırsız hesaplama gücüne dayanıklı bir şekilde gizlilik (anonimlik) sağlama gibi avantajları vardır.

Anahtar Sözcükler: anonimlik, e-oylama, homomorfik sayım, karıştırıcı-ağ, sıralamalı oylama, RSA rassallaştırması, tek geçişli oylama, seçmen izlemesi.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Dr. Nazife Baykal for her support and valuable guidance. I also thank Assist. Prof. Dr. Kemal Bıçakcı for his motivating ideas and helpful discussions.

I would also like to thank my family for giving me the encouragement and all kinds of support during my whole work.

Finally, I would like to thank my colleagues in my company for the support they have given especially during the last period of this thesis work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BCA	Ballot Casting Assurance
BSN	Ballot Serial Number
DRE Machines	Direct Recording by Electronics Machines
E-Voting	Electronic Voting
EVP	Encrypted Vote Product
EVP_{ran}	Randomized Encrypted Vote Product
EVS	Encrypted Vote Sum
OSS	Open Source Software
PAV	Prêt a Voter
PBB	Public Bulletin Board
SBA	Short Ballot Assumption
SCP	Set of Candidate-Primes
SERVE	Secure Electronic Registration and Voting Experiment
STV	Single Transferable Voting
UV	Universal Verifiability
UOCAVA	Uniformed and Overseas Citizens Absentee Voting Act
VP	Vote Product
VP_{ran}	Randomized Vote Product
VS	Vote Sum
VVPAT	Voter-Verified Paper Audit Trail
VVPR	Voter-Verified Paper Records

CHAPTER 1

INTRODUCTION

In recent years, there has been an increasing tendency of carrying out large scale elections by electronic means, which may include the *optical scan devices* and *Direct Recording by Electronics* (DRE) machines and/or computer networks. *Electronic voting* (e-voting) is the general name used to define such systems; extending from *voting at polls recorded by electronic ballot boxes* that may either be counted at polls or cast over closed networks, to *voting cast and recorded over the Internet*.

Election processes in general, whether conducted by electronic means or not, involve diverse groups that have sufficient motive to affect the election results according to their wish. Therefore, the large scale election systems should be *auditable* and if there are disagreements between the *post-election audits* and the actual vote counts, recounts are unavoidable. For the accuracy and transparency of the system, it should also be *universally verifiable*, i.e., any interested third party should be provided with a simple method of checking the final tallying and verifying that only the registered voters cast ballots. The universal verifiability property was firstly defined in [Sako-Kilian-1995] in more general terms as: “In the course of the protocol, the participants broadcast information that allows any voter or interested third party at a later time to verify that the election was performed properly”. An electronic voting system, which is auditable and universally verifiable also necessitates the testability of voting machines; thus, the use of *Open Source Software* (OSS) seems essential [Paul-Tanenbaum-2009].

A large-scale election procedure that is potent enough to replace today's conservative elections is also expected to contribute to democracy by providing new benefits. In addition to the universal verifiability requirement, the state of the art e-voting systems offer each voter the facility of *voter verifiability* without distorting the *anonymity* of the votes, a facility that supports democracy by enhancing the borders of personal rights. The idea of voter verifiability is to endow each voter with the opportunity of verifying that his vote is cast and recorded correctly while preserving the voter privacy, i.e., the anonymity of the vote. This is achieved by using *Public Bulletin Boards* (PBB) and cryptographic tools in general; but there is also a unique example like "Three Voting Protocols: Three Ballot, VAV, and Twin" [Rivest-Smith-2007] that does not use any cryptography. In order to preserve the anonymity of the votes, there are two main methods that heavily use the cryptographic algorithms: i) the mix-nets proposed by Chaum [Chaum-1981], and attacked and/or developed later also in [PfitzmannB&A-1990], [PfitzmannB-1994], [Park-Itoh-Kurosawa-1994] and by many other researchers; ii) the homomorphic tallying introduced by Benaloh, as first described in [Cohen-Fischer-1985], [Benaloh-1986], [Benaloh-Yung-1986], and later featured in [Baudron-Fouque-Pointcheval-Stern-Poupard-2001].

So as to distinguish the concept of voter verifiability (or *individual verifiability*) from universal verifiability, which already takes care of the vote counting phase, voter verifiability is defined as voter's check on the casting and recording phases of his vote [Sako-Kilian-1995] only, and not on the tallying phase. Some researchers [Adida-Neff-2006] also name this property as *Ballot Casting Assurance* (BCA) to make the distinction more clearly. In order that the voting scheme provides ballot casting assurance, voters need not to trust the election officials for the recording of votes, since they can make their own checks.

The e-voting protocols of the 21st century combine all of the above properties to introduce the novel and important concept of *end-to-end system integrity*. Such systems are designed to provide greater assurance that the election outcome is correct than traditional systems; they are, for instance, even more reliable than voting by optical scan machines with post-election auditing [Rivest-2009]. They preserve voter privacy and achieve election integrity, without having to trust the hardware, software or the election officials.

Clearly, the final goal of all e-voting systems is the *simplicity* to gain the confidence of all voters and political parties. Smart use of electronic means and cryptography in the design of the novel e-voting schemes, is anticipated as well to stop the long queues at the polls and increase the number of voters that join the elections; hence, to contribute to democracy over again. To sum up in brief, potential electronic voting systems of the future should undoubtedly,

1) simplify the voting procedure for every voter,
2) make the vote-counting more rapid, accurate, universally verifiable; hence more accurate and reliable,
3) offer each interested voter the facility of verifying in particular that his vote is cast and recorded correctly; while preserving anonymity, and not allowing vote coercion or vote selling even if the voter intends to do so.

1.1 History of Electronic Voting

Among the pioneering electronic means used for e-voting, there are optical scan machines exercised in the US elections since 1962, to tally the ballots. These machines are widely used in today's US elections as well, and the percentage of counties employing them has increased continuously from 0.8% in 1980, to 45.4%, 56.2% and 58.9% in the years 2004, 2006 and 2008 respectively [Election Data Services-2010].

Another electronic means, Direct Recording by Electronics (DRE) machines employed for e-voting are usually the personal computer type of equipment. Touch-screen DRE’s were first used in the US in 1970’s; they run special-purpose voting software, frequently on an operating system like Windows. Ideally, the machines are physically hardened, preventing access to the typical personal computer connectors. DRE’s solve a number of complex operational problems [Adida-2006], like:

- offering ballots in different languages,
- magnifying the screen for voters with vision impairment,
- using a headset that provides auditory feedback,
- simplifying the ballot management by using memory cards instead of paper.

DRE’s, on the other hand, are extensively criticized because they lack a tamper-proof audit-trail. Voting activists and computer scientists are worried that these machines could produce erroneous results, either because of bugs or malicious code, that would go undetected [Landes-2002]. In particular, the worry is that a voter’s choice would be incorrectly recorded at casting time. Such a mistake would be completely untraceable and unrecoverable since the only feedback that a voter obtains is from the voting machine itself. That would damage the auditability, which is one of the main requirements in *electronic elections* (e-elections).

As a remedy to this problem, the Voter-Verified Paper Audit Trail, (VVPAT) was proposed by Mercuri in 1992 [Mercuri-1992]. Once the voter has finished filling out the ballot, the VVPAT-based voting machine prints out an audit of the ballot visible to the voter behind glass. The voter then gets to confirm or cancel his vote. The audit trail effectively short-circuits the machine’s possible mistakes. Ideally, in the case of a recount, the paper trail would be used instead of the electronic record. DRE’s with VVPAT have first appeared in the U.S. voting equipment market in 2003 but their use has grown significantly since November 2006 [Wack-2006]. The percentage of counties employing DRE’s with or without VVPAT has increased from 0.2% in

1980, to 21.7% and 36.3% in the years 2004, 2006 respectively but decreased again to 34.3% in 2008 [Election Data Services-2010], in favor of optical scan devices.

The pioneering e-voting schemes that preserve voter privacy and achieve higher assurance of election integrity, without having to trust the hardware, software, or the election officials were proposed at almost the same time: Chaum's system based on visual cryptography [Chaum-2004], Neff's "MarkPledge" [Neff-2004], Ryan's "Prêt à Voter" [Ryan-2004], and Shubina and Smith's "ElectMe" [Shubina-Smith-2004]. Many other schemes followed shortly, such as "PunchScan" [Chaum-2006], "Prêt à Voter with re-encryption mixes" [Ryan-Schneider-2006], "Scratch&Vote" [Adida-Rivest-2006], "Three Ballot, VAV, Twin" [Rivest-Smith-2007], "Scantegrity" [Chaum-Essex-Carback-Clark-Popoveniuc-Sherman-Vora-2008], "Scantegrity II" [Chaum-Carback-Clark-Essex-Popoveniuc-Rivest-Ryan-Shen-Sherman-08], "Mark Pledge2" [Adida-Neff-2009], "Trustworthy Voting: From Machine to System" [Paul-Tanenbaum-2009], each scheme contributing to its predecessors partially or sometimes significantly.

Internet voting is yet another option to be considered within the context of electronic voting; although it has its own risks of voter coercion and software security. Nevertheless, there are many Internet implementations like 2003 test elections at Vienna University, Austria [Krimmer-2003], legally binding Internet elections of Estonia in 2005, 2007 and 2009, smaller scale elections held by a few cantons in Switzerland since 2003, "Civitas" proposed by the computer scientists at Cornell University [Clarkson-Chong-Myers-2008] and "Helios" [Adida-2008], which is used in the presidential elections at Université catholique de Louvain (UCL) in Belgium [Adida-deMarneffe-Pereira-Quisquater-2009]. A verifiable electronic voting scheme over the Internet is also described in [Li-Hwang-Lai-2009].

Some of the schemes described above are only suitable for the elections, where the voter marks a single choice. However; a few of them are also applicable to preferential voting, where the voter provides a rank-ordering among a given number of candidates. “Prêt à Voter: All-In-One” scheme proposed in 2007, that we call PAV 2007, supports preferential voting as well as single choice . It is developed by a group of researchers [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007], who follow the original ideas in Chaum’s scheme based on visual cryptography [Chaum-2004], and Ryan’s “Prêt à Voter” [Ryan-2004]. Our interest in this work will be more on multiple-winner elections, and preferential voting methods such as the *Single Transferable Voting* (STV).

1.1.1 Electronic Elections in Different Countries

We give illustrative examples of some large scale electronic elections; (i) the U.S. presidential elections in 2000, 2004, 2008 and Senate elections in 2006; which use hybrid tools like optical scan machines or DRE’s in closed networks, in addition to paper ballots, lever machines, punchcards and also ‘voting by mail’ in small percentages; and (ii) those using purely remote voting, like the three e-elections performed in Estonia in 2005, 2007 and 2009 and the election held in Switzerland in 2003. Notable e-voting trials have also been conducted in Austria, Canada, England, Estonia, France, Ireland, Netherlands, New Zealand, Spain and Switzerland. Among them Estonia seems to be pioneering on large scale Internet based e-voting.

(i) Hybrid Elections in the United States

Electronic voting is used in the U.S. for quite a long time, but the most striking experience has been the 2000 Presidential Elections in Florida, where a lot of debate about the reliability of the votes occurred. In the 2000 Presidential Elections, Bush won in Florida only by a margin of 500 votes [Florida-DoS-2000]. There were various complaints such as the misleading of “butterfly ballot” in Broward County, the punchcard system failing to record a number of votes, and more than 50,000 missing ballots [Shapiro-2004]. That was sufficient to show that elections were far from perfect. In 2002, the use of the punchcards was then forbidden by law (HAVA - Help America Vote Act) as a result of the strong debate on the election results.

Election officials were aware of the equipment failures long before 2000 elections [Gumbel-2005]. Nonetheless the failures had not previously been associated in such a near miss election. Many states started to re-evaluate their voting equipment, and searched for more computerized solutions so as to be “not like Florida” [Walton-2004]. These changes raised questions, especially among a number of computer scientists who feared that fully computerized voting would complicate or completely prevent the election verification process [Rubin-2004], [Chasteen-2004].

As a part of U.S. Department of Defense’s Federal Voting Assistance Program, an Internet-based voting system, SERVE, has been started as an experimental project [Jefferson-Rubin-Simons-Wagner-2004]. In the report announced by the U.S. Department of Defense in May 2007, the e-voting plans of the U.S. for 2008-2010 were discussed; that was later criticized by [Jefferson-Rubin-Simons-2007], which concluded that it is impossible to create a secure e-voting system through Internet in those day’s insecure information technology.

The U.S. elections held in 2004, 2006 and 2008 have used the experience gained from the failures of 2000 presidential elections to improve the pre-checks and post-election audits of the system. Many jurisdictions turned from punchcards to electronic voting machines. Unfortunately, most of these machines were not much more than PC’s with touch screens. Some of them were as problematic as punchcard systems, they made recounts impossible; hence some jurisdictions were driven back to paper ballots. The research on e-voting systems and statistical post-auditing of elections became more challenging in order to come across the finest possible solutions before the closest elections; so it started to grow rapidly.

(ii) Remote E-Elections in Estonia and Switzerland

While most of the countries are looking for ways to adopt e-voting for their remote citizens, Estonia is the first country to use electronic voting through Internet for its legally binding local elections in October 2005. The outcome was a success as reported by the Estonian election officials, so the same voting system through

Internet has been used in 2007 and 2009 elections with an increasing percent in usage (105,000 voters, roughly 9.5% of the total voters in 2009).

The system works through use of smart cards that have been distributed to all citizens as an ID card whereas this ID card is used to authenticate the voter. There are three principles in the voting procedure:

- Voters are given a chance to vote many times but only the last vote is counted in the final tally.
- Classical voting overwrites the electronic vote.
- If considerable amount of attack is detected during voting, electoral committee might cancel the e-voting part.

Estonian e-voting system has been analyzed in various academic papers and found to be secure and reliable by Magi [Magi-2007]. On the other hand, a few cantons in Switzerland used a simple e-voting system, where users receive their voter numbers and secret ID's by post and use their votes over the Internet without any extra precaution on security [US-DoD-2007], since 2003. Adaptation of new technology is comparatively small among the Swiss people [Gerlach-Gasser-2009].

1.2 Aim, Contributions and Organization of the Thesis

The aim of this thesis is the investigation of contemporary electronic voting schemes, with special emphasis on voter verifiable preferential voting and homomorphic tallying. We propose a modification in the Single Transferable Voting (STV) method to be applied to large scale elections with electoral barriers. Our proposal prevents the loss of votes and distributes them securely to the second or higher choices of their voters. We present a case study for 2007 Turkish Parliamentary Elections, which demonstrates the advantages of preferential voting for the election systems that have electoral barriers. We also propose three modifications on the "Prêt à Voter: All-In-One" scheme suggested in 2007, to enhance the security of its ballot construction phase. Finally, concentrating on anonymous e-voting, we recommend the RSA algorithm as a candidate for multiplicative homomorphic tallying; present four novel randomization methods for this purpose; one of them being inspired by the ThreeBallot method [Rivest-2006], [Rivest-Smith-2007] that does not use any cryptography. We discuss, criticize and compare the performances of our randomization schemes. We also show how the overall randomization load can be cancelled (before or) after the final decryption to increase the voter capacity of the e-voting system. Our suggestion that employs homomorphic RSA tallying has advantages over El Gamal and Paillier tallying, such as having the least encryption complexity and the strongest anonymity resistant to unlimited computational power.

In Chapter 2, we focus on STV elections and propose a method to be applied to large scale elections, in which political parties whose votes remain below a certain barrier, are eliminated. We then present a case study to demonstrate the effect of preferential voting on the election systems that have electoral barriers. We consider the Turkish Parliamentary Elections held on July 22, 2007 as an example. Section 2.2 is the summary and mathematical formulation of the present tallying strategy ordered by Turkish Parliament Election Law (no. 2839, accepted on June 10, 1983), which utilizes the d'Hondt method.

In Chapter 3, after reviewing the three basic e-voting concepts of voter verifiability, universal verifiability and anonymity; we briefly compare the two applications presented in 2009, with respect to these concepts. Subsequently, we discuss the Prêt à Voter schemes with special emphasis on the "Prêt à Voter: All-In-One" scheme and Paillier encryption. We then present our three modifications in the ballot construction phase, which enhance the security of this scheme.

Chapter 4 is devoted to multiplicative homomorphic tallying to achieve anonymity employing the concept of "vote product" instead of the "vote sum" used by additive homomorphic algorithms. We propose the RSA algorithm as a promising candidate for multiplicative homomorphic tallying and present four new randomization methods for RSA tallying. We also show how the overall randomization term can be cancelled by performing “modular division” in Z_n^*, without bringing any extra load on the vote product. After comparison and critics of our randomization methods, we suggest their joint uses in different cases depending on the number of candidates.

In Chapter 5, we compare our auditable and verifiable e-voting scheme that employs homomorphic RSA tallying with El Gamal, Exponential El Gamal and Paillier tallying, and show that it has the least encryption complexity and the strongest anonymity resistant to unlimited computational power. We then present our simulation results and finish the chapter with an implementation proposal for Turkish Parliamentary Elections.

Chapter 6 is a summary of our contributions and suggestions for further studies.

CHAPTER 2

MODIFIED SINGLE TRANSFERABLE VOTING AND A CASE STUDY FOR ELECTIONS WITH ELECTORAL BARRIERS

In the large-scale elections with electoral barriers, the wasted votes that are given to the candidates that cannot exceed the threshold is a significant problem. Examples of countries using electoral barriers are Belgium (5%, on regional basis), Iceland (5%), Israel (2%), Poland (5%, or 8% for coalitions), Romania (5%), Serbia (5%), Slovenia (4%), Spain (3%) and Turkey (10%). We propose a preferential voting method, for instance the Single Transferable Voting (STV), as a solution to the problem of wasted votes. We recommend a modification on Single Transferable Voting (STV) and adapt it to the Turkish election system that uses the d'Hondt method in tallying. Then, to demonstrate the effect of preferential voting, we present a case study on the Turkish Parliamentary Elections held on July 22, 2007.

In the case study, we search for the answer of the hypothetical question that: "If preferential electronic voting (over the Internet or closed networks accessible from voting booths) were used, how much could the results of 2007 Turkish Parliamentary Elections change?". The inclusion of preferential voting could serve to democracy by distributing the wasted votes securely to the second or other choices of their voters if accurate and auditable management of the accompanying e-voting system was provided.

So, we have applied the d'Hondt's method coerced by the Turkish Election Law to the statistical data gathered in the 2007 Turkish Parliamentary Elections, to predict how the parliament seat distributions of the three winning parties would change if preferential voting were used. After the mathematical formulation of the d'Hondt method with modified STV, we have calculated the number of wasted votes by taking the present details of the tallying strategy within each election region into account. We have then made simulations according to four different and politically unbiased scenarios on the assumed distribution of secondary choices of wasted vote owners in 69 election regions.

We have found drastic changes created by the wasted votes, and tabulated the computed parliament seat distributions under the assumed four scenarios. One of the scenarios results in the most democratic seat distribution, which is in great harmony with the overall percentage of votes. The wide discrepancy between the computed seat distributions under different scenarios makes one think that preferential e-voting may be an efficient means to increase the democracy component in the elections with electoral barriers. However, electronic handling of the voting, casting and tallying phases is crucial for the required speed, security and accuracy of the system, which becomes a little more complicated because of the inclusion of preferential voting.

In Section 2.1, we summarize the STV method an present our modification on the STV so that it can be adapted to the elections with electoral barriers. Section 2.2.1 is the summary and mathematical formulation of the present tallying strategy ordered by Turkish Parliament Election Law (no. 2839, accepted on June 10, 1983), which utilizes the d'Hondt method. In Section 2.2.2, we describe the "modified STV+ d'Hondt" tallying. Section 2.3 presents and interprets the simulated results under four different, politically unbiased scenarios.

2.1 Single Transferable Voting (STV)

Large scale elections having electoral barriers eliminate the political parties, which obtain a vote-percentage below a threshold. People who vote for these parties feel a kind of injustice because of the invalidity and final loss of their votes.

On the other hand, STV elections in which each voter gives a ranked list of preferred political parties may be an excellent democratic solution to the problem of exhausted votes. In the following, we propose a preferential voting application for the elections having such barriers; by modifying the STV method.

2.1.1 Original Single Transferable Voting

Single Transferable Voting is a preferential voting method, in which, voters give a preference ranking of the candidates or political parties rather than indicating a single choice on the ballot as in FPTP elections. The original STV method is a multiple-winner voting method that uses a quota. The quota is determined by dividing the total number of voters to the "number of seats plus one", and then adding one. After the count of the first preferences, the following steps are performed:

1) Any candidate who reaches the quota is elected. His *surplus*, i.e., the excess number of votes over the quota, is distributed to the second choices of the voters who select him as the first in their preferential ordering. The share of each new candidate from the surplus remains proportional to her share in the original distribution.

2) If no one meets the quota, the candidate with the fewest votes is eliminated and his votes are transferred to the second choices of the voters who select him as the first in their preferential ordering.

3) Steps 1 and 2 are repeated until a winner is found for every seat.

Some large scale elections have barriers, which eliminate the political parties that obtain a vote-percentage below a certain threshold. For preventing the problem of wasted votes in such cases, we propose a modification in STV, which is described in the following section. By this modification, no surplus of a winner is distributed; but the wasted votes of the losers are valued. So, the election rule of eliminating the parties below the barrier is preserved, whereas no vote is exhausted.

2.1.2 Our Modification on the STV Method

Modified STV does not use the concept of surplus; i.e., it does not distribute any surplus of the winning candidates, but transfers the votes of the losing parties to the winners. Each voter gives a preference order (of say political parties entering the elections) on the ballot as in the original STV and the tallying phase of the applied voting scheme is modified as follows:

1) In the first evaluation of ballots, only the *first preference* of each voter is counted. If all parties get enough votes to exceed the barrier, the tallying phase terminates.

2) If not all the parties pass the barrier, calling the parties above the threshold *winners* and those below the threshold *losers*, ballot tallying is repeated for all the loser ballots having a loser party as the first choice. The choice on the ballot that is counted this time is the winner party, which is on top of all other winners in the voter’s preference list. The new ballot-counts are added on the votes gained by *winners* in part (1) and the tallying phase terminates.

The above method preserves the election rule of eliminating the parties below the barrier, whereas no vote is exhausted and citizens are satisfied. The method should be accompanied by a secure control mechanism to publicly assure the robustness of the used scheme. To achieve this goal, observer teams assigned by each political party may play critical roles for increasing the robustness and accuracy of the voting scheme.

2.2 Application to the Turkish Election System

Turkish Parliament Election Law contains an electoral barrier of 10%; i.e., political parties that take less than 10% of the overall votes cannot have a member at the parliament. We present the mathematical formulation the Turkish election system and our adaptation of the modified STV method into the system in Section 2.2.1 and Section 2.2.2, respectively.

2.2.1 Tallying Strategy by the d'Hondt Method

In order to discuss the adaptation of the modified STV method to Turkish Parliamentary Elections, the present details of the tallying strategy within the election regions should be taken into consideration. Turkish Parliament Election Law (no. 2839, accepted on June 10, 1983) organizes the tallying strategy as follows (http://www.anayasa.gen.tr/2839sk.htm).

Before the election (item 4 of the law)

i. Let *C* be the number of *C*ities (in the country) and *S* be the number of *S*eats (at the parliament). Each city should have at least one representative in the parliament; so, *C* many of the *S* seats are pre-assigned. The remaining $S-C$ seats are distributed by computing the number of extra representatives e_j for each city as in items (*ii*)– (*v*) given below.

ii. Let *P* be the *P*opulation (of the country according to the last general census), and p_j be the population of the *j*'th city. Total population is then equal to the sum of city *populations, i.e.,*

$$P = \sum_{j=1}^{C} p_j \tag{2.1}$$

.

If *S* is the number of *S*eats and *C* is the number of *C*ities, find the number,

$$a = P/(S-C) \tag{2.2}$$

iii. Divide the population p_j of each city by *a*, to determine the number of extra parliamentary members to be assigned to that city. Notice that if p_j / a were an integer for all cities, the procedure would terminate and all of the remaining $S - C$ seats would be distributed; because, dividing equation (2.1) by *a*, one obtains $\frac{P}{a} = \sum_{j=1}^{C} \frac{p_j}{a} = S - C$. However, practically p_j is never an integer multiple of *a*, hence the greatest integer contained in p_j / a is assigned as the *initial* value of the extra parliament members e_j to be elected by the voters of the *j*'th city.

iv. For all the cities *j*=1,…,*C*, the remainders of p_j / a are put into descending order. The remaining parts of the $S - C$ seats, which cannot be distributed in part (*iii*), are distributed according to this order. For each city, the number of extra parliament members e_j is finalized by adding 0 or 1 to its initial value found in part (*iii*). Together with the initial seat assigned to each city in part (*i*), the number of representatives to be elected by the voters of the *j*'th city is $s_j = e_j + 1$. The total number of seats is then

$$S = \sum_{j=1}^{C} s_j . \tag{2.3}$$

v. A city with assigned number of representatives $s_j \leq 18$ is counted as a single election region; if $18 < s_j \leq 35$, the city is divided into two, and if $35 < s_j$, it is divided into three election regions by following the principles described in the Turkish Parliament Election Law.

After the election (item 33 and item 34 of the law)

i. If total votes of a political party cannot exceed 10% of the valid votes used nationwide, that party cannot be represented at the parliament (say *B* (out of *A*) parties exceed the 10% barrier).

ii. Let the *k*'th one among *B* parties get v_{jk} many votes, where *k*=1,…, *B*, in the election region *j* with assigned number of representatives s_j. Number of votes v_{jk} of each party is divided by 1, 2, 3, up to s_j, and the numbers

$$\left\{v_{j1}, \frac{v_{j1}}{2}, \ldots, \frac{v_{j1}}{s_j}, v_{j2}, , \frac{v_{j2}}{2}, \ldots, \frac{v_{j2}}{s_j}, \ldots, v_{jB}, \ldots, \frac{v_{jB}}{s_j}\right\}$$

together with the votes given to the independent candidates are put into descending order. Finally, the first s_j parties (or independent candidates) in the list are selected as the winners of the *j*'th election region (according to the d'Hondt method).

Example: Say *A* many parties join the elections and *B* (<*A*) of them exceed the 10% threshold. We want to find the winners in the *j*'th election region, for which the assigned number of representatives is $s_j = 5$. Calling the number of votes given to *B* parties in this election region $\{v_{j1}, v_{j2}, \ldots, v_{jB}\}$, and assuming that there is one independent candidate who has taken *v* many votes, let's suppose that the list in descending order starts by $\{v_{j2},\ v_{j2}/2,\ v_{jB},\ v,\ v_{j2}/3,\ v_{j2}/4,\ v_{j3}, \ldots\}$. Taking the first 5 elements of this list, Party 2 that has v_{j2} many votes wins the three seats; Party *B* and the independent candidate win the remaining two seats.

2.2.2 Adaptation of the Modified STV Method

Adaptation of the modified STV method to Turkish parliamentary elections is not complicated, if the number of votes v_{jk} obtained by each of the *B* parties exceeding the 10% barrier can be updated automatically after determining the parties which stay below the threshold. In Chapter 3, we will discuss the homomorphic property of the Paillier encryption scheme used by "Prêt à Voter: All-In-One" [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007] that allows the computation of voter's preferences in a single attempt. So, we think that the number of updated votes, $v_{jk} \mid E$, conditioned on the event *E*={Some given parties are below the 10%

threshold}, can be calculated efficiently if “Prêt à Voter: All-In-One” is used. In the ensemble of ballots whose first preferences are the losing parties, the software ignores all losing parties simultaneously and adds the new counts on v_{jk} to obtain the updated number of votes, $v_{jk} \mid E$. More specifically, if A many parties join the elections and B ($<A$) of them exceed the 10% threshold nationwide; let’s call the parties above the threshold 1,2, …, B and those below the threshold B+1, …, A, without loss of generality. Hence, the event E is defined as

E={Parties B+1, …, A are below the 10% threshold}.

Number of votes in the j’th election region, $\{v_{j1}, v_{j2}, ..., v_{jB}\}$ is now to be updated by transferring the number of votes $\{v_{j,B+1}, v_{j,B+2}, ..., v_{jA}\}$ to parties 1,2, …, B; such that the sum of the updated votes, $\sum_{k=1}^{B}(v_{jk} \mid E)$, conditioned on the event E, is equal to the total number of votes, $\sum_{k=1}^{A} v_{jk}$, before knowing the event E. The corresponding software performs the following algorithm:

1) Collect all ballot results whose first choice is a losing party, into a single file F;

2) Ignoring all preferences given to parties B+1, …, A; find the first choice of each ballot in F that belongs to the set of parties {1, 2, …, B};

3) In F, compute the total count of ballots $\{\hat{v}_{j1}, \hat{v}_{j2}, ..., \hat{v}_{jB}\}$ for each party in the set {1, 2, …, B}, such that

$$\sum_{k=1}^{B} \hat{v}_{jk} = \sum_{k=B+1}^{A} v_{jk} \text{ ;}$$

4) Find the updated number of votes by $(v_{jk} \mid E) = v_{jk} + \hat{v}_{jk}$, for k=1,…,B;

5) Stop.

The final check is $\sum_{k=1}^{B}(v_{jk} \mid E) = \sum_{k=1}^{A} v_{jk}$. The resulting updated numbers $v_{jk} \mid E$ are now to be substituted for the previous values v_{jk} to adapt the STV method.

2.3 Turkish Parliamentary Elections Held in 2007

In the Turkish Parliamentary Elections held on July 22, 2007, there were 14 political parties joining the elections and three of them became winners by exceeding the 10% barrier. The amount of "wasted votes" used for the 11 losing political parties and independent candidates was around 15%. In this section, using the vote counts of the Turkish Parliamentary Elections held on July 22, 2007, we compute how the number of parliament seats would be affected according to different scenarios on the distribution of preferential votes. We assume that the votes used for the winning parties that exceed the electoral barrier of 10% are only processed once. However; the votes used for the losing parties are reprocessed to find out the winning party with the highest rank in voter's preference list. We also assume that the new ballot-counts are added on the votes gained by winners in the first count. Below, we firstly summarize the actual election results and secondly, we present our simulations according to four scenarios on the distribution of secondary or higher preferences.

2.3.1 Actual Results

As can be observed from Table 2.1, three parties, AKP, CHP and MHP are the winners, since they exceed the 10% barrier. In Table 2.2, we demonstrate the percentages of the parliament seats distributed by the d'Hondt method as described in the Election Law (no. 2839, accepted on June 10, 1983) together with the actual vote percentages. We also include in Table 2.2, the discrepancies between seat and vote percentages, which becomes as high as 15.53%.

Table 2.1 Results of 2007 Turkish Parliamentary Elections (first three and the last columns are taken from: Turkish Official Gazette, no: 26598, July 30, 2007) arranged in descending order of vote counts.

Political Party	**Number of votes**	**Percentage**	**Percentage**	**Parliament Seat Ratio**	**Parl. Seats**
ADALET VE KALKINMA PARTİSİ (**AKP**)	16,327,291	46.58%	46.58%	62.11%	341
CUMHURİYET HALK PARTİSİ (**CHP**)	7,317,808	20.88%	20.88%	20.40%	112
MİLLİYETÇİ HAREKET PARTİSİ **MHP**)	5,001,869	14.27%	14.27%	12.75%	70
DEMOKRAT PARTİ (**DP**)	1,898,873	5.42%	18.27%		
INDEPENDENT CANDIDATES	1,835,486	5.24%		4.74%	26
GENÇ PARTİ (**GP**)	1,064,871	3.04%			
SAADET PARTİSİ (**SP**)	820,289	2.34%			
BAĞIMSIZ TÜRKİYE PARTİSİ (**BTP**)	182,095	0.52%			
HALKIN YÜKSELİŞİ PARTİSİ (**HYP**)	179,010	0.51%			
İŞÇİ PARTİSİ (**İP**)	128,148	0.37%			
AYDINLIK TÜRKİYE PARTİSİ (**ATP**)	100,982	0.29%			
TÜRKİYE KOMÜNİST PARTİSİ (**TKP**)	79,258	0.23%			
ÖZGÜRLÜK VE DAYANIŞMA PARTİSİ (**ÖDP**)	52,055	0.15%			
LİBERAL DEMOKRAT PARTİ (**LDP**)	35,364	0.10%			
EMEK PARTİSİ (**EP**)	26,292	0.08%			

Table 2.2 Vote and parliament seat percentages for the winning parties of the 2007 Turkish Parliamentary Elections in 85 election regions (the whole country).

Percentages of:	**AKP**	**CHP**	**MHP**	**Sum of 3 columns**
Votes in the 2007 TP elections	46.58%	20.88%	14.27%	81.73%
Seats in the 2007 TP elections	62.11%	20.40%	12.75%	95.26%
Discrepancy (=Seat% − Vote%)	15.53%	−0.48%	−1.52%	13.53%

The last row of Table 2.2 shows how much the "proportional representation" principle of democracy is twisted as a result of tallying by the d'Hondt method. As for a closer comparison between the winning parties, we show the relative vote and seat percentages among the three winners in Table 2.3 that we obtain by dividing the vote and seat percentages in the first two rows of Table 2.2, to the corresponding last column element. The last column of Table 2.3, found by adding the previous three columns, explains its difference from Table 2.2; because the percentages are now computed within the set of three winning parties only, and the last column adds up either to 100% or 0%. The last row of Table 2.3 shows how much the d'Hondt method favors the first winning party with respect to the second and the third parties.

Table 2.3 Relative percentage of votes and parliament seats among the three winning parties of the 2007 Turkish Parliamentary Elections in 85 regions (all country).

Percentages of:	**AKP**	**CHP**	**MHP**	**Sum of 3 columns**
Votes in the 2007 TP elections	56.99%	25.55%	17.46%	100%
Seats in the 2007 TP elections	65.2%	21.4%	13.4%	100%
Discrepancy (=Seat% − Vote%)	8.2%	−4.1%	−4.1%	0%

2.3.2 Simulated "Modified STV+d'Hondt" Results

Using the vote counts of the Turkish Parliament elections held on July 22, 2007, in which there were three winning parties, we have predicted how the parliament seat distribution of the winner parties would change, if voters used preferential votes and the 'votes of the voters who voted for the loser parties' were transferred by the modified STV method as explained in Section 2.2.2.

We have computed the "modified STV+d'Hondt" results under four different scenarios on the secondary or higher choice distribution of wasted votes. We have found upper bounds of the "modified STV+d'Hondt parliament seat distribution" in 2007 elections, by considering 3 extreme scenarios each assuming: 'only one of the three winning parties takes all of the secondary votes'. The last one, Scenario 4 assumes equal secondary vote distribution for all the winning parties.

Simulation program separately works for each scenario and each election region with number of representatives s_j as follows:

1. Computes the sum of wasted votes from actual counts in the "election region *j*",

2. Under a given scenario, adds the corresponding percentage of wasted votes to the initial votes of the winner parties, to find the 'corrected vote counts',
3. In array A, ranks all corrected vote counts in descending order,
4. Finds one half, one third, one fourth,…, $1/s_j$ th of all the votes in array A,
5. In array B, arranges all numbers found at steps 3 and 4, in descending order,
6. Chooses the largest s_j numbers of array B, as parliament members of the election region j.

Whenever an independent candidate wins in an election region; the scenarios for the STV part of the tallying phase become unrealistic, hence meaningless to simulate. Therefore, we have restricted our analysis to 69 election regions, where there is no independent candidate among the winning parliament members, out of the total of 85 election regions. Discarded 16 regions with independent parliament members are: Bitlis, Diyarbakır, Hakkari, İstanbul (1), İstanbul (3), Mardin, Muş, Rize, Siirt, Sivas, Şanlıurfa, Tunceli, Van, Batman, Şırnak, Iğdır. We calculate the wasted vote percentage in the remaining 69 election regions as 16.19% by dividing the number of votes used for the loser parties that remain below the electoral barrier (4,556,611) to the total number of votes (28,141,705) in these 69 regions.

Total number of the elected parliament members within these 69 election regions of interest is 433 (instead of 550 seats corresponding to the whole country with 85 election regions). Distribution of the 433 parliament seats among the three winning parties is as shown in Figure 2.1 Dividing the number of seats in Figure 2.1 by the total number of 433 seats, one computes the seat percentages 63%, 22% and 15%, which are quite different from the actual vote percentages of 56%, 25% and 19%, respectively. This difference is partly a result of the d'Hondt method. However, with the preferential voting assumption used in this study (especially as in Scenario 4), our computations show that seat and vote percentages approach to each other

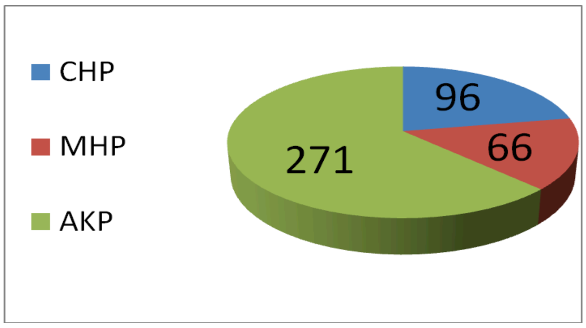

Figure 2.1 Distribution of 433 parliament seats to the three winning parties, in the 69 election regions that don't have any independent parliament members.

We tabulate the vote and seat percentages in the 69 election regions of the 2007 Turkish Parliamentary (TP) elections in Table 2.4 and emphasize the 'vote and seat percentage discrepancy' in the last row.

Table 2.4 Relative percentages of votes and parliament seats among the winning parties of the 2007 Turkish Parliamentary Elections in 69 regions.

Percentages of:	**AKP**	**CHP**	**MHP**	**Sum of 3 columns**
Votes in the 2007 TP elections	56%	25%	19%	100%
Seats in the 2007 TP elections	63%	22%	15%	100%
Discrepancy (=Seat% − Vote%)	7%	−3%	−4%	0%

In order to determine the maximum changes of "number of parliament seats" between the actual results and simulated "modified STV+d'Hondt" results; we firstly consider the 3 extreme scenarios of "only one winning party takes all secondary votes"; Scenario 1: All secondary votes to AKP, Scenario 2: All secondary votes to CHP, Scenario 3: All secondary votes to MHP. Then, in a more realistic scenario, Scenario 4, we assume that secondary votes are equally distributed.

Scenario 1:

If all secondary choices were used for AKP; we compute the seat distribution {AKP, CHP, MHP}= {312, 73, 48} shown in the first row of Table 2.5, instead of the actual distribution {271, 96, 66}. So under the assumption of Scenario 1, AKP would gain +41 more parliament members, 23 seats coming from CHP, and 18 from MHP as shown in the row of Table 2.6 corresponding to Scenario 1.

Scenario 2:

If all secondary choices were used for CHP; the new seat distribution would be {223, 161, 49} as shown in Table 2.5, so CHP would gain +65 more parliament members, 48 from AKP, 17 from MHP as shown in the row of Table 2.6 corresponding to Scenario 2.

Scenario 3:

If all secondary choices were used for MHP; seat distribution would be as shown in Table 2.5 and MHP would gain +77 more parliament members, 51 from AKP and 26 from CHP as shown in Table 2.6.

Table 2.5 Number of parliament seats according to four different scenarios, of the three winning parties in 69 regions, predicted by modified STV+d'Hondt method.

Scenarios	AKP	CHP	MHP
Scenario 1: If all secondary votes were used for AKP	312	73	48
Scenario 2: If all secondary votes were used for CHP	223	161	49
Scenario 3: If all secondary votes were used for MHP	220	70	143
Scenario 4: If they were equally distributed	254	104	75
Actual number of Parliament seats	271	96	66

Scenario 4:

Finally, we consider a more realistic scenario for the distribution of secondary votes such as equal distribution to all winning parties. If secondary choices were equally distributed; seat distribution would be as shown in Table 2.5 and CHP would gain 8 and MHP would gain 9 parliament seats whereas AKP would lose 17 of its seats, as shown in Table 2.6.

Table 2.6 Differences that would occur according to four different scenarios, in the present number of parliament seats of the three winning parties in 69 regions, predicted by modified STV+d'Hondt method.

Scenarios	**AKP**	**CHP**	**MHP**
Scenario 1: If all secondary votes were used for AKP	+41	−23	−18
Scenario 2: If all secondary votes were used for CHP	−48	+65	−17
Scenario 3: If all secondary votes were used for MHP	−51	−26	+77
Scenario 4: If they were equally distributed	−17	+8	+9

Predicted percentages of parliament seats under each of these four scenarios are computed and tabulated in Table 2.7. Last two rows of Table 2.7 show that the equal distribution assumption of Scenario 4, for the secondary votes of the wasted vote owners, yields a seat percentage very close to the actual vote percentage.

Table 2.7 Predicted relative percentage of parliament seats among the three winning parties according to four different scenarios in 69 regions.

Relative Percentages of:	**AKP**	**CHP**	**MHP**
Parliament seats for Scenario 1	72%	17%	11%
Parliament seats for Scenario 2	52%	37%	11%
Parliament seats for Scenario 3	51%	16%	33%
Parliament seats for Scenario 4	**59%**	**24%**	**17%**
Votes in 2007 elections	**56%**	**25%**	**19%**

The analysis of simulations held on the actual results of the 69 election regions in 2007 Turkish Parliament elections under the first 3 scenarios reveal that, if the voters were given the chance of preferential voting, the election results could change drastically. Equi-distributed secondary votes assumption of Scenario 4 yields a seat distribution very close to the actual vote distribution; which is a pleasant result in terms of democracy, as shown in Table 2.8.

It is worth noting that we have tried to make politically unbiased assumptions on the distribution of secondary votes and been interested in the 3 simplest scenarios that give upper limits of the parliament member distributions under mSTV+d'Hondt tallying. Scenario 4 is also unbiased, since it gives equal chance to all winner parties.

Table 2.8 Discrepancies between seat and vote percentages of 2007 Turkish Parliamentary Elections in 69 regions.

Parliament member percentages	AKP	CHP	MHP
Presently differ from vote percentage by:	7%	−3%	−4%
Predictions of Scenario 4 differ from vote percentage by:	3%	−1%	−2%

We have intentionally avoided more realistic scenarios that take the social and political assets of each election region into account. Because our aim is to show the possible limits of change in parliament member distributions between "d'Hondt only" and "modified STV+d'Hondt" cases, while keeping the neutrality of the study. The wasted vote percentage in these 69 election regions is 16.19%. Unfortunately, the drastic change predicted by the modified STV+d'Hondt method in the first three scenarios can be interpreted as a measure of the amount of "democracy lost in wasted votes", which could nevertheless be gained back by asking the second and higher preferences of the voters.

For the implementation of preferential elections, we think that electronic voting is indispensable, since it is a perfect medium for vote transferring as multiple recounts and exhaustive comparison of the possible outcomes of the votes may be required. A voter verifiable method such as "Prêt a Voter: All-In-One" scheme [Xia-Schneider-

Heather-Ryan-Lundin-Peel-Howard-2007] may be a good choice on such accounts because of its suitability for STV. Heather gives the security measures for such an application [Heather-2007] and the improved version of Prêt a Voter with application to STV can be found in [Xia-Schneider-Heather-Traore-2008].

2.4 Conclusion

We have formulated the “modified STV+d’Hondt” method according to the Turkish Parliament Election Law (no. 2839, accepted on June 10, 1983) by taking into consideration the details of the present tallying phase within each election region. We have then made some simulations using the vote counts of 2007 Turkish Parliamentary Elections, under four simple but politically unbiased scenarios on the distribution of secondary or higher vote preferences. Our computations made by the “modified STV+d’Hondt” method disclose that, if only the voters were given the chance of preferential voting, the election results could change quite drastically. One of our scenarios, in which we assume that the secondary choices of the wasted votes are distributed uniformly among the winning parties, results in the most democratic seat distribution, i.e., proportional representation, which is in great harmony with the overall percentage of votes.

In conclusion, we think that tallying methods disregarding the effect of wasted votes should not be preferable in today’s world where the concept of democracy is very significant. A transferable voting system like the “modified STV+d’Hondt” as in our simulations seems to have the power of efficiently cancelling the democratically unpleasant effect of wasted votes; especially for the elections with high barriers.

Electronic voting is indispensable for preferential elections, since it is a perfect medium for vote transferring methods as multiple recounts and exhaustive comparison of the possible outcomes of the votes may be required. A voter verifiable method such as PAV 2007 to be described in Chapter 3 may be a good choice on such accounts because of its suitability for STV.

We have presented the work explained in Section 2.2 together with the original contents of Chapter 3 at ECEG'2009, "9th European Conference on e-Government", in London, UK, on June 29-30, 2009 [Yücel-Baykal-2009]. Our simulation results using the vote counts of July 22, 2007 Turkish Parliamentary Elections, and making the assumption of preferential voting under the four politically unbiased scenarios of Chapter 2 is presented at the 4th Information Security and Cryptology Conference, ISC'10 in May 2010 [Yücel-Baykal-2010-a].

CHAPTER 3

“PRET A VOTER” E-VOTING SCHEMES AND VOTER VERIFIABILITY

After a brief review of basic e-voting concepts such as universal verifiability, voter verifiability and anonymity in Section 3.1; we briefly compare the two recent e-voting applications with regard to these properties: the first one implementing a system with end-to-end integration [Adida-deMarneffe-Pereira-Quisquater-2009], and the second one emphasizing the use of open source software [Paul-Tanenbaum-2009] for auditable elections. In Section 3.2, we review the voter and universal verifiable Prêt à Voter (PAV) schemes, which are first proposed in 2004. They provide voting receipts without any threat of voter coercion and ballot-selling and achieve anonymity through mix networks. Our special emphasis will be on the ‘Prêt à Voter: All-In-One’ scheme proposed by [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007]. This scheme utilizes the Paillier encryption [Paillier-1999] but its new version [Xia-Schneider-Heather-Traore-2008] employs the El Gamal encryption [El Gamal-1985]. Both of the mentioned e-voting schemes are very suitable for the STV applications discussed in Chapter 2 and this is why they attract our particular interest. In Section 3.2.2, we present a brief and comparative review of PAV 2005, PAV 2006, and PAV 2007 schemes. In Section 3.2.3 , we discuss the cryptographic details of the ‘Prêt à Voter: All-In-One’ (PAV 2007) scheme. Section

3.3 presents and discusses our three modifications in the ballot construction phase of the PAV 2007 scheme. We summarize our conclusions in Section 3.4.

3.1 Fundamental E-Voting Concepts and Two Applications

Electronic voting over the Internet or closed networks accessible from voting booths has several advantages over paper based voting, especially in terms of voter verifiability. Arising new protocols for electronic voting are competing with each other and solving the earlier encountered problems such as the confliction between anonymity and voter verifiability.

Voter verifiability is a concept which does not exist in traditional paper-based elections but becomes an important issue of democracy in the electronic world. The idea is to endow each voter with the facility of verifying that his vote is counted. The first level of such a verification could be to verify that his vote is cast and recorded correctly and the second level is whether his vote is counted (or tallied) correctly. More rigorous definition of voter verifiability includes this second step within the concept of *universal verifiability*. The check mechanism for the correct recording of the vote can be provided by means of a voting receipt. On the other hand, whenever one has a receipt that serves to check the correct recording of the vote, it can also be used as the proof for the content of the vote used in the election. This may in turn lead to voter coercion and ballot-selling. Hence, previous versions of the electronic voting protocols have avoided giving receipts to the voters, and introduced the concept of receipt-freeness as an integral part of the voting system. On the other hand, the psychology of voters, who are traditionally used to paper based voting is much more on the side of having something touchable, like a paper in hand, rather than completely relying on the electronic media.

'Prêt à Voter' schemes are electronic voting schemes, which respond to this psychological need of having the receipt. More importantly, the way that the receipt is constructed provides voter verifiability up to the second level, i.e., each voter can verify that his vote is cast correctly, by means of the receipt that does not tell anything about the content of the vote to anybody except for the voter himself. The second level of verifiability, i.e., the check of correct tallying of each received vote is managed by the universal verifiability of the overall system. In other words, the robustness, correctness and dependability of the overall system is provable. Prêt à Voter schemes have their origins in the schemes proposed by Chaum [Chaum-2004] and Ryan [Ryan-2004].

Among different types of voting systems depending on the needs of the particular election, FPTP (First Past The Post), STV (Single Transferable Voting), Condorcet and Borda Count elections are the main ones to be mentioned. The 'Prêt à Voter' scheme proposed by Chaum, Ryan and Schneider [Chaum-Ryan-Schneider-2005], [Ryan-Peacock-2005] in 2005, PAV 2005, and its enhanced form PAV 2006 [Ryan-Schneider-2006] are easily implemented for FPTP elections; however, they may be complicated for application to STV elections. On the other hand, 'Prêt à Voter: All-In-One' scheme [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007] that we will call PAV 2007 throughout this thesis, and its new version [Xia-Schneider-Heather-Traore-2008] that employs the El Gamal encryption solve the problem of handling the STV elections efficiently.

3.1.1 Three Basic Concepts: UV, BCA and Anonymity

Election process in general involves diverse groups that have sufficient motive to affect the election results according to their wish. So, recounts in large scale elections are unavoidable if there are disagreements on the vote counts. A universally verifiable [Sako-Kilian-1995] e-voting system offers any interested individual the right and facility of controlling the accuracy of the overall tallying procedure as well as the eligible voter lists. An auditable system also necessitates the testability of voting machines; hence, the use of open source software (OSS) seems essential [Anderson-2008].

In addition to the universal verifiability requirement, contemporary e-voting systems also offer the facility of voter verifiability (or individual verifiability [Sako-Kilian-1995]), which supports democracy by enhancing the borders of personal rights as compared to the conventional voting systems. The property of voter verifiability seems to provide the joint check of three notions: whether an individual vote is i) cast as intended by its voter, ii) recorded at the tallying office as cast, and iii) tallied as recorded. However, the third notion can be omitted for two reasons: i) to chase a single vote up to the point of tallying is not meaningful after the check of correct recording; ii) correct count of all recorded votes is a concern of universal verifiability (UV) as well, which offers all interested citizens the opportunity and right to control the correctness of the overall tallying process. In other words, UV provides full public control on the vote-count and eligibility of voters.

Individual (or voter) verifiability, is also called the *Ballot Casting Assurance* (BCA) by other researchers [Adida-Neff-2006], who combine the notions that a vote is "cast as intended" and "recorded as cast" into the concept of BCA. Ballot casting assurance also requires a detailed assist to the voter, by keeping the voting machine responsible of proving to the voter in zero knowledge that his vote is recorded correctly. In order that the voting scheme provides BCA, voters do not need to trust the election officials for the recording of votes, since they can confidently make their own checks. Combination of these two properties, namely UV and BCA, produces an o*pen-audit* voting system.

Almost all present-day e-voting schemes, such as the Prêt a Voter [Ryan-Peacock-2005], [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007], the Punchscan [Chaum-2006], [Fisher-Carback-Sherman-2006], the Scantegrity [Chaum-Essex-Carback-Clark-Popoveniuc-Sherman-Vora-2008], the Scantegrity II [Chaum-Carback-Clark-Essex-Popoveniuc-Rivest-Ryan-Shen-Sherman-2008], Scratch& Vote [Adida-Rivest-2006] or Helios [Adida-2008] are open-audit, that conform to the above mentioned concepts of universal verifiability (UV) and ballot casting assurance (BCA). These schemes use similar tools to achieve the UV and BCA efficiently; the main idea being to provide the voter with a post-voting encrypted receipt; which can only prove to himself the way that he used his vote, and to nobody else. Encrypted votes of all eligible voters are announced on public bulletin boards (PBB's), so BCA is satisfied since anybody can check that his vote is recorded. UV is also satisfied since i) anybody can control whether a voter is among the eligible ones, together with this voter's encrypted (hence protected) vote, ii) the measures for the confirmation of tallying by any interested party are entirely taken.

On the other hand; these schemes may differ in the ways to accomplish the *anonymity*, that is, voter privacy concerns. There are two main methods in providing the anonymity of the votes: i) Ballots can be encrypted, shuffled, re-encrypted and re-shuffled by mix-nets or ii) Anonymity is achieved by utilizing homomorphic tallying, where a special public key algorithm is needed for decrypting an aggregate of encrypted ballots, without decrypting any ballot separately. We summarize one of the two anonymity tools, mix-nets in Section 3.1.2; and give an example that uses the second tool of homomorphic tallying in Section 3.1.3. However, in the rest of this chapter, we consider schemes that use mix-nets for anonymity and deal with homomorphic tallying more extensively in Chapter 4.

To provide examples as regards the concepts of UV and BCA discussed above and the concept of anonymity that will be dealt with in the remaining part of the study, we compare in brief, the two randomly chosen e-voting applications presented in 2009: the first scheme implementing Internet voting with end-to-end integration [Adida-deMarneffe-Pereira-Quisquater-2009] and the second one emphasizing the use of OSS [Paul-Tanenbaum-2009] on the DRE's for voting at the polls in Sections 3.1.3 and 3.1.4.

3.1.2 Mix Networks (Mix-Nets) for Anonymity

As mentioned before, there are two main methods in providing the anonymity of the votes; the earlier idea being the mix-nets, composed of a set of mix servers (also called layers) cascaded to each other. Ballots can be encrypted and shuffled by mix-nets as first suggested in 1981 by Chaum [Chaum-1981], who used RSA onions with random padding. To produce the encrypted onion, plaintext is repeatedly encrypted using a different public key and random padding at each layer. The first Chaumian network was attacked by [PfitzmannB&A-1990] using the multiplicative homomorphism of raw RSA and independent randomness of the padding. Later, [Park-Itoh-Kurosawa-1994] proposed the first re-encryption mix-net, where each mix server re-randomizes the ciphertexts with fresh randomization values, which is again attacked by [PfitzmannB-1994]. The first universally verifiable mix-net was proposed by Sako and Kilian in 1995 [Sako-Kilian-1995] based on the techniques given in [Park-Itoh-Kurosawa-1994]. Sako and Kilian's work was the first mix-net to provide a proof of correct mixing that any observer can verify. PAV systems that we have described in Chapter 2 also employ such re-encrpytion mix-nets.

E-voting applications require robust mix-nets to generate shuffled lists of encrypted messages. Correctness constraint is so stringent that, even if all mix servers are corrupt, there is no tolerance to the loss or altering of any message. Most robust mix-net protocols make use of a PBB, which is ideally expected to record all postings such that any interested observer can check.

An alternative way of achieving anonymity is to utilize additive homomorphic tallying as first proposed by Benaloh (previously Cohen) ([Cohen-Fischer-1985], [Benaloh-Yung-1986]) for decrypting an aggregate of encrypted ballots, without decrypting any ballot separately. For this purpose one needs to use a special public key algorithm in tallying, which possesses homomorphic additivity, like the Exponential El Gamal and Paillier algorithms.

3.1.3 Internet Voting Held at the Université catholique de Louvain with Additive Homomorphic Tallying for Anonymity

In 2008, Université catholique de Louvain (UCL) in Belgium decided to open up its presidential elections in March 2009 to all university members. Since there are 25,000 eligible voters at UCL, who are more educated than a random group chosen from the society, an election over the Internet seemed pretty feasible. The researchers of UCL collaborated with Ben Adida from Harvard University, the writer of Helios 1.0 [Adida-2008], which is one of the pioneering web-based open-audit voting systems that provides BCA and UV. The researchers then presented their work [Adida–deMarneffe–Pereira–Quisquater-2009] during the EVT'2009 E-Voting Technologies Workshop held in Montreal in August 2009 **(**and they took the *Best Paper* award).

Helios 2.0 software was developed in accordance with the needs of the UCL election; and released as OSS. Instead of the mix-nets used by Helios 1.0, anonymity of the scheme was achieved by homomorphic tallying in Helios 2.0. Exponential El Gamal encryption was used since it is found to be easier to implement than Paillier encryption. Main advantage of Exponential El Gamal over Paillier is its suitability for distributed decryption with joint key generation. One of the most difficult parts in the application (because of the lack of uniformity in the technical expertise of trustees) was the joint generation of El-Gamal public keys by multiple independent trustees, which were then combined by multiplication. Decryption for tallying was done partially by each trustee. It was observed by the authors that it is more important for trustees to develop their own key generation code than their own verification code. According to the authors, this sounds quite logical since verification can be performed many times after the elections, while safe key generation and partial decryption must be done correctly in a short time.

After the voting phase, the PBB that contains all the voter identifications and vote hashes was frozen: a signed receipt was published for each vote, together with a signed version of the PBB content. After publication of the signed receipts, a full day was devoted to the PBB audits. Voters were invited to consult the PBB, to produce a new ballot and introduce their objections to the election commission in case of disagreement with the signed data.

Around 4000 voters joined the elections; almost 30% of them checked their receipts on the PBB. There were very small number of complaints and none of them was about the essence of the system.

We consider the presented implementation as one of the best trials of Internet voting and cannot find anything to criticize about it. We also conclude that UV, BCA and anonymity are very efficiently solved by this scheme.

3.1.4 Paul and Tanenbaums's E-Voting System as an Example with Weak Anonymity

"Trustworthy Voting: From Machine to System" is introduced in May 2009 by some researchers at the Vrije University in Amsterdam [Paul-Tanenbaum-2009]. It is an e-voting system for use on the voting machines at the polls, and concentrates on the audits of OSS by means of the Trusted Platform Module (TPM), which allows the verification of the voting machine in real time, by demonstrating that the machine runs the open source software that it is supposed to run. The main system goals are integrity, traceability and simplicity to gain the confidence of all voters and political parties.

The scheme uses open source software, and offers 'attestation' which lets anyone verify that the published software is running on the published software. Attestation is achieved by means of computing the hash of the published software and comparing it with the one running on the machine.

In order to compute the hash over machine's software in a reliable manner, Paul and Tanenbaum suggest the use of TPM (Trusted Platform Module), which is a device that is already part of many modern PCs and has a special instruction, called *skinit*, that can be used for software attestation. The overall voting system consists of 9 steps and uses public key pairs defined for three cases, the first pair per polling location (or per machine), the second pair per voter for ballot signing, and the third optional pair is created per person who is interested in software attestation. 9 phases of the voting system, first 4 of them to be performed before the election, items 5 to 8 at the election day and the last one being after the election, are:

1) Generation and distribution of key pairs per polls,
2) Creation of voter registration records,
3) Mail proof of registration to voters,
4) Preparation of voting machines,
5) Assembling two halves of the private key of the polls just before the polls open,
6) Checking-in voters at the polls; giving the voting tokens for acceptable registration cards,
7) Voting by voter's card, token and password (after optional attestation) and casting,
8) Tabulation of votes at the polls; reporting the results by telephone followed by escorted transportation of the results to the county registrar,
9) Publicizing the results at the county registrar for voter verification, after the attestation of the software by any interested party.

Our critics on this scheme is two-fold: 1) We think that although the audits for the OSS is considered in detail and voter registration procedure is described with ultimate care, the scheme falls beyond its third main goal: Simplicity cannot be provided with so many public key pairs and the crucial expectation from each voter of remembering his password. If every voter is persistent and skillful enough to use his two pairs (one is optional) of public/secret keys and correctly remember his password, the scheme can work very securely; hence remaining two of the main system goals: integrity and traceability can be achieved. 2) However, we observe that the scheme is somewhat weak in providing voter privacy; since the votes are announced at the PBB in plain form, together with some random numbers for the check of each voter. Even if only the random numbers (without the votes) were announced as suggested by the authors, in case "the state decides that vote selling is a bigger problem than election tampering", we think that there is no obvious measure of this scheme that guarantees the anonymity of individual votes against the state.

So, we conclude that although UV and BCA is solved efficiently by this scheme, anonymity is an unsettled problem.

3.2 'Prêt à Voter' Schemes

We firstly describe the common properties of the Prêt à Voter schemes, PAV 2005 [Chaum-Ryan-Schneider-2005], [Ryan-Peacock-2005], followed by PAV 2006 [Ryan-Schneider-2006], PAV 2007 [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007] and its new version [Xia-Schneider-Heather-Traore-2008] that employs the El Gamal encryption in Section 3.2.1; then give a comparative discussion in Section 3.2.2. Section 3.2.3 is devoted to the explanation of the cryptographic core used in 'Prêt à Voter: All-In-One' (i.e., PAV 2007) scheme [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007], with vital details which cannot be found in the original reference.

3.2.1 Common Properties of the 'Prêt à Voter' Schemes

Prêt à Voter schemes, PAV 2005, PAV 2006 and PAV 2007 have their origins in the two-sheet visual cryptographic scheme proposed by Chaum [Chaum-2004] and two-column ballot suggested by Ryan [Ryan-2004]. All Prêt à Voter schemes are voter-verifiable methods, which provide voting receipts without any threat of coercion and ballot-selling. Ballots contain two columns as shown in Figure 3.1, say the left column including the candidate names in random order (or alphabetical order simply rotated by a random cyclic shift, say 2 downward shifts as in Figure 3.1) and the right column having an encrypted number called *'onion'* that is prepared as a function of the random order of candidates on this ballot. In the voting booth, voter marks his vote on the right column; and the left column of the ballot on which the candidates are tabulated is destroyed after voting, as a part of the voting phase.

The provided receipt includes only the right column that shows the marked vote and the onion; hence it is meaningless to anybody except the voter himself, who has seen the order of the candidates before the removal of the left column of the ballot.

Demet	X
Efe	
Ayşegül	
Binnur	
Cihan	
	8HbWs6

Figure 3.1 Ballot form used by original Prêt a Voter schemes (after voting but before removal of the left column).

Anonymity, i.e., voter privacy, of Prêt à Voter schemes is accomplished by means of the mix-nets, which are proposed by Chaum [Chaum-1981].There are *k* tellers (or clerks) of the mix-net, each one having two sets of a public/private key pair. Ballots are prepared before the elections; the secret number, which indicates the candidate order on the left column of the ballot is encrypted one-after-another with 2*k* public keys of *k* tellers to form the onion. To decrypt this onion in the tallying phase, contribution of each teller is indispensable, because each layer of the encrypted onion can only be decrypted by one of the private keys of the corresponding teller. In order to preserve anonymity and break any linkage between the voter and decrypted vote, each teller randomly permutes the output list of ballots before submitting it to the next teller.

Voter verifiability [Sako-Kilian-1995], [Adida-Neff-2006] of Prêt à Voter schemes is achieved by means of encrypted receipts and public bulletin boards (PBB), which have universally accessible memory, and provide public communication, such that an election authority can write secured, unalterable and undeletable information on it and any other party can read. The receipt containing the right column with voter's mark and the onion will be scanned after voting phase, and published onto the bulletin board by election authorities. Any voter can check for the existence and integrity of his receipt on the bulletin board and make an objection whenever any problem occurs. Perhaps the most attractive part of the Prêt à Voter schemes is that, although the receipts are publicized, nobody except the voter himself is able to understand the content of the vote, unless the onion is decrypted properly. However, decryption of the onion is distributed cleverly to *k* tellers, who also have to randomly shuffle the ballot lists that they receive, before submitting to the next teller; so that the link between the initial voter and the final tallied vote cannot be tracked unless all tellers are corrupted.

3.2.2 Differences Among ‘Prêt à Voter’ Schemes

Improvements brought by PAV 2006 over PAV 2005 are,

i) preparation of ballot forms with two onions, the new-left onion on the left column being encrypted by the public key of the voting machine and decrypted by its private key whenever the voter casts his vote,

ii) distributed generation of ballot forms, to enhance the security of ballot generation phase,

iii) on demand-printing of ballots to resist the chain voting attack reported in [Ryan- Peacock-2005],

iv) size-independence of onion from the number of tellers,

v) separation of shuffle and decryption phases to increase robustness, by first shuffling the received votes by a re-encryption mix-net as suggested in [Neff-2001].

The main improvement brought by PAV 2007 over PAV 2006 is its adjustment according to the needs of preferential elections, such as STV or Condorcet elections. Since each voter makes a ranking of the candidates in preferential elections, the order of the ballot candidate list needs to be totally randomized as in Figure 3.2, instead of simply cyclically shifting the same alphabetically ordered list on each ballot.

Ayşegül	$\boxed{5}$ dBOpTf
Efe	$\boxed{3}$ 66rdMv
Demet	$\boxed{1}$ Abc123
Binnur	$\boxed{2}$ 7YJLfN
Cihan	$\boxed{4}$ Vs68Hb

Figure 3.2 Ballot form used by “Prêt a Voter: All-In-One” (PAV 2007) scheme (after voting but before removal of the left column).

The implementation of previous PAV 2005 and PAV 2006 schemes may be complicated with this constraint, as compared to the implementation of the approval elections. PAV 2007 takes care of this problem by treating the ballot as a whole in the ballot tallying phase. It makes use of the Paillier encryption [Paillier-1999] and exploits its homomorphic property in absorbing all ranked choices of the ballot within a single encrypted onion. Ballots are then shuffled by a re-encryption mix-net composed of many tellers (or clerks). In later versions of Prêt à Voter schemes, El Gamal [Xia-Schneider-Heather-Traore-2008] and Paillier [Ryan-2008] ciphers are used respectively.

3.2.3 Encryption and Decryption by Paillier Cryptosystem

Brief description of Paillier cryptosystem [Paillier-1999], [Paillier-Pointcheval-1999] is needed at this point for clarifying our contribution to "Prêt à Voter: All-In-One" (PAV 2007) scheme. The Paillier algorithm (see Section 4.1.1) has the additive homomorphic property; that is, an encryption of m_1+m_2 can be obtained from any encryption of m_1 and m_2, as

$$E(m_1, r_1)E(m_2, r_2) \equiv E(m_1+m_2, r_1r_2). \tag{3.1}$$

In (3.1), the ciphertext $c= E(m, r)$ stands for the encrypted form of the plaintext m,

$$c = E(m, r) = g^m r^n \pmod{n^2}, \tag{3.2}$$

r is chosen at random, n and g are fixed public elements defined by

$n = pq$, the modulus where p and q are large primes

$g \in Z^*_{n^2}$, and equal to 1 mod n

So, the pair (n, g) is the public key, and the pair (p, q) or equivalently λ = LCM $(p-1, q-1)$ is the private key of the Paillier's algorithm. As a consequence of (3.1), it is clear that

$$E(m, r)^k \equiv E(km, r^k). \qquad (3.3)$$

Ballot Construction:

In Figure 3.2, the numbers on the rightmost are obtained as a result of successive encryptions performed by the clerks of the mix network. The order {1, 5, 4, 2, 3} on the ballot shown in Figure 3.2 of the candidates {1: Ayşegül, 2: Binnur, 3: Cihan, 4: Demet, 5: Efe} is chosen randomly by the first clerk. First clerk also picks up the random numbers $r_1, r_2, \ldots, r_5$ and prepares the encrypted numbers $c_1, c_2, \ldots, c_5$ showing the candidate placed at each row, using (3.2) and the public key (n, g). More specifically, the encrypted number that the first clerk prepares for the j'th row is

$$c_j = E(M^i, r_j) = g^{M^i} r_j^n \pmod{n^2} \qquad (3.4)$$

where M is any integer greater than the number, v, of candidates, and i=1,2,…,v shows the alphabetical order of the candidate corresponding to that row. Hence, for the example of Figure 3.2, the candidate order {1, 5, 4, 2, 3} is reflected directly to the exponents of M as follows

$$c_1 = E(M^1, r_1),\quad c_2 = E(M^5, r_2),\quad c_3 = E(M^4, r_3),\ c_4 = E(M^2, r_4),\quad c_5 = E(M^3, r_5).$$

Each successive clerk of the mix network re-encrypts the numbers $c_1, c_2, \ldots, c_5$ by multiplying them with the n'th power of a random number t, so that the new value of the ciphertext becomes

$$\tilde{c}_j = c_j = g^{M^i} r_j^n t^n = E(M^i, r_j t). \qquad (3.5)$$

Equation (3.5) shows that, the plaintext M^i remains untouched while the random numbers picked by the first clerk are each time multiplied by a different random number t; so that the first clerk, the only person who knows the candidate order, cannot trace the ballot. Since $\tilde{c}_j$ value of the j'th row keeps the message M^i in encrypted form, the number i showing the alphabetical order of the candidate sitting at the j'th row is always preserved and not affected by increasing number of re-encryptions. The numbers on the rightmost of the ballot in Figure 3.2 are those $\tilde{c}_j$ values obtained by the last clerk at the end of the ballot construction chain.

Ballot Casting:

The voter casts his vote by ranking, i.e., filling up the numbers 1 to v in the right hand column; tears the ballot apart, destroys the left part and keeps the right part as the receipt after being scanned by the election authority at the voting booth. The scanned receipt is also announced at the PBB for further checks demanded by the voter or any other party.

Ballot Tallying:

After the ballot is ranked by the voter, a single onion for each ballot having v candidates is calculated as follows:

$$E(m,r) = \prod_{i=1}^{v} E(M^i, r_i)^{k_i} = \prod_{i=1}^{v} \tilde{c}_i^{k_i}\,, \tag{3.6}$$

where k_i's indicate the voter's ranking corresponding to the candidate, who is the i'th one in alphabetical order. In the decryption of this onion, homomorphism of the Paillier algorithm leads to a very useful result:

$$\prod_{i=1}^{v} E(M^i, r_i)^{k_i} = E(\sum_{i=1}^{v} k_i M^i, \prod_{i=1}^{v} r_i^{k_i})\,. \tag{3.7}$$

The authority who has the private key (p, q) or equivalently λ = LCM (p-1, q-1) corresponding to the public key pair (n, g), extracts the useful message m in $E(m, r)$ given by (3.6). Because of the homomorphic property in (3.7),

$$m = \sum_{i=1}^{v} k_i M^i \text{ and since } M > v, \tag{3.8}$$

retrieval of all choices of the ballot becomes possible. Because from (3.8) it is clear that (m / M^v) equals k_v, the {remainder of m/ M^v} divided by $M^{v\text{-}1}$ equals $k_{v\text{-}1}$; and the remainder after dividing by $M^{v\text{-}1}$, again divided by $M^{v\text{-}2}$ gives $k_{v\text{-}2}$ and so on.

Example: In Figure 3.2, the candidate order {1, 5, 4, 2, 3} during the ballot construction is reflected to the subscript i of k_i's, as $\{k_1, k_5, k_4, k_2, k_3\}$ and the voter's choice on the ballot shows that k_1=5, k_5=3, k_4=1, k_2=2, k_3=4.

If M (>v=5) is chosen as 7, the following useful message m is obtained from (3.8):

$$m = \sum_{i=1}^{v} k_i M^i = 5\times 7^1 + 3\times 7^5 + 1\times 7^4 + 2\times 7^2 + 4\times 7^3.$$

So, m divided by 7^5 gives k_5=3, hence we understand that the alphabetically 5'th candidate Efe is the third in the list of the voter,
Rem{m/7^5}/7^4 gives k_4=1, which shows that the alphabetically 4'th candidate Demet is the first choice of the voter,
Rem{Rem{m/7^5}/7^4}/7^3= k_3 = 4 , so Cihan is his 4'th choice,
Rem{Rem{Rem{m/7^5}/7^4}/7^3}/7^2 = k_2 = 2, so Binnur is the 2'nd, and finally, Ayşegül is the last choice of the voter since

Rem{Rem{Rem{Rem{m/7^5}/7^4}/7^3}/7^2}/7 = k_1 = 5

3.3 Our Proposals to Enhance the Security of "Prêt à Voter: All-In-One" (PAV 2007) Scheme

The tallying phase of the PAV 2007 scheme [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007], which is inherently very suitable for preferential elections can be easily modified according to the STV method suggested above. We strongly predict that such an e-voting scheme may become an indispensable alternative for paper-based elections, if the security of the overall system is enhanced. The weakest point in ballot-construction phase of the PAV 2007 scheme seems to be the over-dependence on the first clerk, who decides on the random candidate order of the ballots. He chooses this random order, and encrypts it by using random seeds for each row of the ballot and the public keys of the election authority, who is responsible for tallying the votes. The re-encryptions performed by the following clerks in the network has the purpose of obscuring the path that the ballot follows.

So, our first proposal to enhance the security of the system is to hold the first clerk in the chain more responsible of the encryption he performs, by including his digital signature as a part of the encrypted information, which may be checked whenever a need occurs. Such a modification increases the robustness of the scheme since any corrupted behavior is known to be traceable even in the future.

The second modification that we propose to enhance the security is in the re-encryption equation (3.5) that is used by the other clerks of mix network,

$$\widetilde{c}_j = c_j t_j^n = g^{M^i} r_j^n t_j^n = E(M^i, r_j t_j) \tag{3.9}$$

in which we change the random number t by a row dependent random number t_j, so that each clerk in the network generates v (number of candidates and ballot rows) random numbers for each ballot, rather than a single one. The use of v random numbers by each clerk (or teller), will make the path more invisible and difficult to catch by the first clerk, who generates the crucial random ordering of the candidates on the ballot.

Finally, we propose the insertion of a Ballot Serial Number (BSN) to be produced whenever a ballot is needed, instead of PAV 2007's Ballot Onion. BSN is used after voting, for the purpose of systematical search of the receipt on the bulletin board. BSN does not contain any information about the candidate order of the ballot, because candidate names are clearly seen on the left column before voting and destroyed after voting.

3.4 Conclusion

After discussing the concepts of universal verifiability (UV), ballot casting assurance (BCA) and anonymity; we briefly review the mix-nets, and Prêt à Voter schemes. The voter-verifiable e-voting scheme, 'Prêt à Voter: All-In-One' proposed in 2007 (so called PAV 2007 in our work) [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007], which supports single transferable voting (STV) seems to be a very suitable option for the elections having electoral barriers; so we suggest a modified STV implementation that utilizes PAV 2007 for such elections. To increase the security and the robustness of the overall system, we modify PAV-2007 by proposing three security enhancing modifications in its ballot construction phase: 1) digital signature of the first clerk in the mix-net, 2) different random numbers for each row of the ballot, 3) ballot serial number. The analysis we present for the cryptographic part of PAV 2007 is more detailed and comprehensive than given in the original paper written by Xia, Schneider, Heather, Ryan, Lundin, Peel and Howard, in 2007.

We presented the work discussed in Chapter 3 at the "3rd Information Security and Cryptology Conference-ISC'08", held in Ankara, on December 25-27, 2008 [Yücel-Baykal-2008].

In Sections 2.1.2 and 2.2.2, we have formulated the "modified STV+d'Hondt" method according to the Turkish Parliament Election Law (no. 2839, accepted on June 10, 1983) by taking into consideration the details of the present tallying phase within each election region. We presented a combination of Chapter 3 and Sections 2.1.2 & 2.2.2 at ECEG'2009, "9th European Conference on e-Government", in London, on June 29-30, 2009 [Yücel-Baykal-2009].

CHAPTER 4

HOMOMORPHIC RSA TALLYING WITH PROPER RANDOMIZATION

As mentioned in the previous chapter, one of the main methods in providing the anonymity of the votes is homomorphic tallying, where a special public key algorithm is needed for decrypting an aggregate of encrypted ballots, without decrypting any ballot separately. In Section 4.1, we briefly review the additive or multiplicative homomorphism of public key encryption schemes, such as the RSA, El Gamal, Exponential El Gamal and Paillier algorithms. We describe how "voter and universal verifiable e-voting with homomorphic RSA tallying" becomes possible by utilizing the multiplicative homomorphic property of RSA in Section 4.2. We propose RSA tallying by assigning a prime number for each candidate on electronic ballots, and using the unique prime factorization of the vote product (*VP*) to find the vote counts. For the implementation of homomorphic tallying, RSA needs to be randomized; but the usual RSA randomization method of padding random bits does not work for this case. So, in Section 4.3, we propose four novel randomization methods for homomorphic RSA tallying; which have excellent potential for practical applications. Section 4.4 is devoted to the discussion of practical considerations about these randomization schemes, such as the dependence of the voter set size to the RSA modulus size, transmission of the RSA randomization factors from voting booths to the final tallying office and their blind cancellation. We then compare our four randomization methods in the same section. Cancellation of the overall randomization factor without bringing any extra load on the RSA modulus is discussed in Appendix A.

4.1 Homomorphic Tallying for Anonymity

Some public key encryption algorithms enjoy the excellent property of *additive homomorphism*, which makes homomorphic tallying possible, and a useful tool for providing anonymity. Homomorphic tallying does not require separate decryption of ballots; instead, all encrypted ballot values are multiplied and the vote sum is decrypted jointly as first described in [Cohen-Fischer-1985], [Benaloh-Yung-1986], and later featured in [Baudron-Fouque-Pointcheval-Stern-Poupard-2001]. Subsequently, all votes remain anonymous without any need for mix-nets. Multi candidate homomorphic systems have been first investigated in [Cramer-Franklin-Schoenmakers-Yung-1996], further discussed and studied in [Cramer-Gennaro-Schoenmakers-1997]. Specific details of the homomorphic multi-counter are given in [Katz-Myers-Ostrovsky-2001]. There is extensive literature on additive homomorphic tallying, but multiplicative homomorphic tallying is only considered in [Peng-Aditya-Boyd-Dawson-Lee-2004] for the El Gamal encryption. To our knowledge, there is no reference in literature to the RSA algorithm as a candidate for homomorphic tallying. Our proposal in Section 4.2 that employs unique prime factorization is general and it is suitable for the RSA algorithm as well as El Gamal.

4.1.1 Homomorphic Public Key Algorithms

In a public-key cryptographic system [Diffie-Hellman-1976], each user of the public key algorithm has to own a key pair, {*public key*, *secret key*}. The mathematical relation between these keys is such that it is almost impossible to find the secret key from the knowledge of the public key; whereas, the other way is trivial. Because of this unique mathematical relation between the keys; any message encrypted with the public key (that everbody knows) can be decrypted with the secret key (which is possessed only by the owner of that secret key). This explains how privacy of the secret key owner is granted by encryption. Conversely; if the message is encrypted (by the secret key owner) with the secret key, then it can be decrypted by anybody who knows the public key. This idea forms the background of a digital signature; which serves not only to the authentication of its sender but also to the integrity of the transmitted message. The enclosure of the concept of message integrity makes digital signatures more powerful than conventional hand-written signatures.

To exhibit the homomorphic properties of some public key algorithms like the RSA, El Gamal, which possess the property of *homomorphic multiplication*, and Paillier, Exponential El Gamal, having the property of *homomorphic addition*; we start by their brief descriptions in chronological order.

For each algorithm summarized below, E refers to the specific encryption function and D refers to the corresponding decryption function; such that $E(D(x))=D(E(x))=x$. If encryption is used for privacy purposes (that is, not for signing to prove identity); then the sender uses the public key *pk* of the receiver and the intended receiver uses the corresponding secret key *sk*. So, for privacy applications, the encryption operation E depends on *pk*, but the corresponding decryption is a function of the secret key *sk*. The algorithm descriptions below are all given for this case, considering privacy instead of authentication. However, conversion to the case of authentication is trivial just by interchanging the roles of *pk* and *sk*. In each case, the plaintext message m is encrypted to obtain the ciphertext c.

<u>RSA Algorithm</u> [Rivest-Shamir-Adleman-1978]

Two large primes p and q are chosen by the user, $n = pq$ and $\Phi(pq) = (p-1)(q-1)$, where Φ is called Euler's totient function (that satisfies $m^{\Phi(n)}(\text{mod } n)=1$ for any m in Z_n^*, n is called the modulus, it is also a part of the public key);

Public key, *pk*: (n, e), where e and $\Phi(n)$ are co-prime and $1 < e < \Phi(n)$,

Secret key, *sk*: d, where $ed = 1 \pmod{\Phi(n)}$

Encryption: $c = E(m) = m^e \pmod{n}$ (4.1)

Decryption: $m = D(c) = c^d \pmod{n}$

Proof: $c^d \bmod n = m^{ed} \pmod{n} = m^{k\Phi(n)+1} \pmod{n} = (m^{\Phi(n)})^k m \pmod{n} = m \pmod{n}$.

El Gamal Algorithm [El Gamal-1985]

p is a known large prime, q is a known large prime factor of $p - 1$, g is the known generator of a q-order subgroup of Z_p^*, r is chosen randomly by the user for each encryption.

Public key, *pk*: $y = g^x \pmod{p}$

Secret key, *sk*: x, selected by the user randomly such that $x \in Z_q^*$

Encryption: $c = E(m) = (\alpha, \beta) = (g^r, m\, y^r) \pmod{p}$ (4.2)

Decryption: $m = D(c) = [\beta / \alpha^x] \pmod{p}$

Proof: $m = D(c) = [\beta / \alpha^x] \pmod{p} = [m\, y^r / g^{rx}] \pmod{p} = [m\, g^{xr} / g^{rx}] \pmod{p}$

Exponential El Gamal Algorithm

If the plaintext m in the El Gamal algorithm, is changed to g^m, the rest is the same as the El Gamal algorithm, except that one needs to take $\log_g$ in decryption. So, p is a known large prime, q is a known large prime factor of $p - 1$, g is the known generator of a q-order subgroup of Z_p^*, r is chosen randomly by the user for each encryption.

Public key, *pk*: $y = g^x \pmod{p}$

Secret key, *sk*: x, selected by the user randomly such that $x \in Z_q^*$

Encryption: $c = E(m) = (\alpha, \beta) = (g^r, g^m y^r) \pmod{p}$ (4.3)

Decryption: $m = D(c) = [\ \log_g (\beta / \alpha^x)\ ] \pmod p$

<u>Paillier Algorithm</u> [Paillier-1999]

Two large primes p and q are chosen, $n = pq$, $\lambda = \mathrm{lcm}\,(p-1, q-1)$, $L(x) = (x-1)/n$.

Public key, *pk*: (n, g), where the order of $g \in Z^*_{n^2}$ is a multiple of n.

Secret key, *sk*: λ (or equivalently (p, q)).

Encryption: $c = E(m) = g^m r^n \pmod{n^2}$ for a random $r \in Z^*_{n^2}$ (4.4)

Decryption: $m = D(c) = [\ L(c^{\lambda} \pmod{n^2}) / L(g^{\lambda} \pmod{n^2})\ ] \pmod n$.

4.1.2 Additive versus Multiplicative Homomorphism

In this section, we briefly review the concepts of additive and multiplicative homomorphism of the algorithms given in Section 4.1.1 in chronological order. Our focus is on the application of these homomorphic properties to the tallying phase of e-voting schemes. Researchers, who propose homomorphic tallying as a means for anonymity, are mostly interested in public key algorithms such as Paillier and Exponential El Gamal that possess homomorphic addition property. We study homomorphic multiplication and propose RSA as a candidate for homomorphic tallying.

After briefly exhibiting the multiplicative homomorphism of RSA and El Gamal, we re-demonstrate additive homomorphism of Paillier and Exponential El Gamal for completeness, by using the encryption equations (4.1) to (4.4) given in the previous section. We note that the convention in naming these properties as "additive" or "multiplicative", refers to the operation performed on the plaintexts and not on the ciphertexts. The encryption algorithm is called additive homomorphic, if the encryption converts the <u>message sum</u> (m_1+m_2) to the ciphertext product (c_1c_2). The algorithm is called multiplicative homomorphic, if the encryption converts the <u>message product</u> (m_1m_2) to the ciphertext product (c_1c_2).

RSA encryption (4.1) has the homomorphic multiplication property; because, if two ciphertexts $c_1 = E(m_1) = m_1^d \pmod n$ and $c_2 = E(m_2) = m_2^d \pmod n$ given by (4.1) are multiplied, one obtains $c_1\, c_2 = m_1^d\, m_2^d \pmod n = (m_1\, m_2)^d \pmod n$, which is the RSA encryption of the product (m_1m_2) of the plaintexts. So,

$$c_1\, c_2 = E(m_1)E(m_2) = E(m_1m_2), \tag{4.5}$$

The property defined by (4.5) is called *homomorphic multiplication*, since the encryption converts the product of plaintexts (m_1m_2) to the product of ciphertexts. Using RSA's decryption function D on (4.5), one can recover (m_1m_2) as

$$D(c_1\, c_2) = D(E(m_1)E(m_2)) = m_1m_2. \tag{4.6}$$

El Gamal encryption (4.2) exhibits homomorphic multiplication property just like RSA; because, if $c_1 = E(m_1, r_1) = (g^{r_1}, m_1y^{r_1})$ and $c_2 = E(m_2, r_2) = (g^{r_2}, m_2y^{r_2})$ given by (4.2) are multiplied, one obtains $c_1\, c_2 = (g^{r_1}\, g^{r_2}, m_1m_2y^{r_1}\, y^{r_2}) = (g^{r_1+r_2}, m_1m_2y^{r_1+r_2})$. So, messages are multiplied (and random numbers are added) as

$$c_1\, c_2 = E(m_1, r_1)E(m_2, r_2) = E(m_1m_2, r_1+r_2). \tag{4.7}$$

Using El Gamal's decryption function D on (4.7), one can then recover (m_1m_2) as

$$D(c_1\, c_2) = D(E(m_1, r_1)E(m_2, r_2)) = (m_1m_2, r_1+r_2). \tag{4.8}$$

Exponential El Gamal encryption (4.3) that uses the plaintext as the exponent of some integer, differs from the El Gamal encryption slightly. The fact that the plaintext is used in the exponent makes the algorithm additive homomorphic; and changes (4.7) and (4.8) as follows

$$c_1\, c_2 = E(m_1, r_1)E(m_2, r_2) = E(m_1+m_2, r_1+r_2). \tag{4.9}$$

$$D(c_1\, c_2) = D(E(m_1, r_1)E(m_2, r_2)) = (m_1+m_2, r_1+r_2). \tag{4.10}$$

The property defined by (4.9) is called *homomorphic addition*, since the encryption converts the sum of plaintexts (m_1+m_2) to the product of ciphertexts ($c_1\, c_2$). Using the corresponding decryption function D on (4.9), one can recover (m_1+m_2) as in (4.10).

Paillier encryption (4.4) is additive homomorphic as mentioned in Section 3.2.3. That is, if two ciphertexts $c_1 = E(m_1) = g^{m_1}r_1^{\,n}$ and $c_2 = E(m_2) = g^{m_2}r_2^{\,n}$ given by (4.4) are multiplied, one obtains $c_1c_2 = g^{m_1}r_1^{\,n}\, g^{m_2}r_2^{\,n} = g^{m_1+\,m_2}(r_1r_2)^n$, which is the Paillier encryption of the sum (m_1+m_2) of the plaintexts with random number r_1r_2, so

$$c_1\, c_2 = E(m_1,\, r_1)E(m_2,\, r_2) = E(m_1+m_2,\, r_1r_2). \qquad (4.11)$$

Using the corresponding Paillier's decryption function D on (4.11), one can recover (m_1+m_2) as

$$D(c_1\, c_2) = D(E(m_1,\, r_1)E(m_2,\, r_2)) = (m_1+m_2,\, r_1r_2). \qquad (4.12)$$

The main accomplishment of the last two algorithms is the decryption of the sum (m_1+m_2) from the product of encryptions, $E(m_1,\, r_1)E(m_2,\, r_2)$, without decrypting any individual plaintext m_1 or m_2. Homomorphic tallying employs this property in order to directly count the total number of votes while preserving privacy and anonymity of each ballot. On the other hand, RSA and El Gamal algorithms are considered to be not suitable for homomorphic tallying since they don't have the homomorphic addition property.

4.2 Prime Factorization for Multiplicative Homomorphic Tallying

As discussed earlier in this chapter, anonymity of votes is granted by homomorphic tallying as it doesn't require separate decryption of ballots. Instead, all encrypted ballot values are multiplied and decrypted jointly to find out the total vote sum corresponding to each candidate, as first described in [Cohen-Fischer-1985], [Benaloh-Yung-1986], and later featured in [Baudron-Fouque-Pointcheval-Stern-Poupard-2001]. Recent e-voting schemes that employ homomorphic tallying either prefer the Paillier or the Exponential El Gamal encryptions because of the *additive homomorphism* of these algorithms. Specific details of the homomorphic additive multi-counter are given in [Katz-Myers-Ostrovsky-2001].

RSA and El Gamal algorithms lack the property of homomorphic addition; however, they have the homomorphic multiplication property. In this part of our work, we show how the homomorphic multiplication property can be employed for homomorphic tallying; provided that ballots are prepared properly, so as to assign a specific prime number for each vote given to that candidate. In [Peng-Aditya-Boyd-Dawson-Lee-2004], *multiplicative homomorphic* tallying by El Gamal encryption is considered. Our description of multiplicative homomorphic tallying in this section is more general and works for any multiplicative homomorphic encryption scheme. However, our focus is on homomorphic RSA tallying, which is firstly proposed in this work.

The absence of random parameters in the RSA algorithm is a disadvantage, especially within the context of e-voting, where each voter uses the same public key of the tallying authority and the number of messages to be encrypted is equal to the limited number of candidates. Because of the lack of randomization, a specific plaintext m always yields the same ciphertext under a given key, so the probability of collisions among ciphertexts increases. In many applications, by adding random padding bits to the plaintext, one can avoid this drawback. However; this is not possible for homomorphic tallying with RSA. In Section 4.3, we discuss why random padding bits do not work for homomorphic tallying, and propose different randomization solutions.

Let us now consider a C-candidate election, where the number of voters is N and the number of votes used for candidates $\{1, 2, \ldots, C\}$ are equal to $\{v_1, v_2, \ldots, v_C\}$ respectively, so their sum is $N = \sum_{i=1}^{C} v_i$. The set of C prime numbers $\{p_1, p_2, \ldots, p_C\}$ are associated with the set of candidates $\{1, 2, \ldots, C\}$, to be used in ballot construction. We will call the set of prime numbers associated with candidates, the "Set of Candidate-Primes", SCP=$\{p_1, p_2,\ldots, p_C\}$. Electronic ballots are prepared so as to record a vote given to candidate j as $E(p_j)$, i.e., the encrypted form of the corresponding prime number p_j, where the encryption algorithm E is required to possess multiplicative homomorphism. Homomorphic tallying procedure first computes the *Encrypted Vote Product* (*EVP*) of all encrypted votes

$$EVP = \prod_{l=1}^{v_1} E(p_1) \prod_{l=1}^{v_2} E(p_2) \ \ldots \ \prod_{l=1}^{v_C} E(p_C), \text{ over } N \text{ voters.} \tag{4.13}$$

The first individual product in *EVP* is of the form $\prod_{l=1}^{v_1} E(p_1)$, and because of the homomorphic multiplication property of the RSA (or El Gamal) algorithms it is equal to

$$\prod_{l=1}^{v_1} E(p_1) = E(\prod_{l=1}^{v_1} p_1), \tag{4.14}$$

using (4.5) (or (4.7)). Since the product of v_1 many p_1's is simply $\prod_{l=1}^{v_1} p_1 = p_1^{v_1}$, the right hand side of (4.14) is equal to $E(p_1^{v_1})$. Similarly, the second term of *EVP* is found as $E(\prod_{l=1}^{v_2} p_2) = E(p_2^{v_2})$, and the last term becomes $E(\prod_{l=1}^{v_C} p_C) = E(p_C^{v_C})$. Substituting all these terms into (4.13), *EVP* is computed as

$$\begin{aligned} EVP = \prod_{l=1}^{v_1} E(p_1) \prod_{l=1}^{v_2} E(p_2) \ \ldots \ \prod_{l=1}^{v_C} E(p_n) &= E(p_1^{v_1}) E(p_2^{v_2}) \ \ldots \ E(p_C^{v_C}) \\ &= \prod_{l=1}^{C} E(p_i^{v_i}) \end{aligned} \tag{4.15}$$

Using the homomorphic multiplication property given by (4.5) (or (4.7)) once more, we obtain

$$EVP = \prod_{i=1}^{v_1} E(p_1) \prod_{i=1}^{v_2} E(p_2) \ \dots \ \prod_{i=1}^{v_C} E(p_C) = \prod_{i=1}^{C} E(p_i^{v_i}) = E(\prod_{i=1}^{C} p_i^{v_i}). \qquad (4.16)$$

Finally, decryption of *EVP* by (4.6) of the RSA algorithm (or by (4.8) of the El Gamal algorithm), produces the *Vote Product*, $VP = \prod_{i=1}^{C} p_i^{v_i}$,

$$VP = D\{\prod_{i=1}^{v_1} E(p_1) \prod_{i=1}^{v_2} E(p_2) \ \dots \ \prod_{i=1}^{v_C} E(p_n)\} = D\{E(\prod_{i=1}^{C} p_i^{v_i})\} = \prod_{i=1}^{C} p_i^{v_i}, \qquad (4.17)$$

which is composed of only prime numbers; therefore can uniquely be factorized to evaluate the vote counts $\{v_1, v_2, \dots, v_C\}$ corresponding to each candidate.

Ballot Casting Assurance (Voter Verifiability): Any vote for candidate j is recorded, and given to the voter as the receipt $E(p_j)$ to be announced together with voter's identity that can be checked on the PBB. Hence the voter verifiability of the system is achieved. (In Section 4.3, we discuss why the receipt needs to be randomized. Then we propose different randomizations for the RSA algorithm, all using multiplication by random numbers r, to obtain receipts of the form $E(rp_j)$.)

Universal Verifiability: Not only the talliers, but also all interested individuals are able to compute the encrypted vote product *EVP*, by multiplying the receipts on the PBB. Everybody can then encrypt the announced *VP* by using the public key of the tallying office and compare it with the product of all receipts on the PBB. Hence the universal verifiability of the system is also accomplished.

Example: In an election region with N=3,000 voters, number of candidates that join the elections is C=5. The ballot forms are prepared such that a vote to be used for Candidate 1 is recorded as $E(p_1)$, where p_1 is the prime number associated with

Candidate 1, and any vote given to Candidate 2 is recorded as $E(p_2)$ and so on. The prime numbers are chosen as SCP=$\{p_1, p_2, p_3, p_4, p_5\}$=$\{2, 3, 5, 7, 11\}$. After the election terminates, all votes are recorded on the public bulletin board (PBB) in encrypted form, so that any interested voter can check that his encrypted vote is recorded correctly. If RSA algorithm is used, randomization of the plaintext is necessary to securely preserve the voter privacy.

Talliers compute the product of all encrypted votes announced on the PBB, and find the encrypted vote product *EVP*. In order to decrypt the *EVP*, the secret key of the tallying authority is necessary; therefore (4.17) can only be computed by the tallying authority. After obtaining the vote product $VP = \prod_{i=1}^{C} p_i^{v_i}$, all that needs to be done is to find the numbers $\{v_1, v_2, v_3, v_4, v_5\}$ by consecutively dividing the *VP* to each prime number. For instance, starting with p_1=2, assume that *VP* is divisible by 2 exactly 1,128 times, but the 1,129'th division is not possible; which shows that v_1=1,128.

Then $VP'=VP/2^{1128}$ is successively divided by p_2=3 until no more division is possible. The number of times that division by 3 can be performed gives the vote count of Candidate 2, say as v_2=324. The maximum number of possible divisions of $VP''=VP/2^{1128}3^{324}$ by p_3=5 is equal to v_3 and the tallying algorithm continues by dividing the sequential values VP''', VP'''' to p_4=7 and p_5=11 to obtain the vote counts v_4 and v_5. After the announcement of the vote counts $\{v_1, v_2, v_3, v_4, v_5\}$, any interested individual can compute the vote product $VP = \prod_{i=1}^{C} p_i^{v_i} = 2^{1128}\, 3^{324}\, 5^{463}\, 7^{931}\, 11^{134}$, and check the result by encrypting *VP* with the public key of the tallying authority. If $E(VP)=EVP$, the tallying is confirmed. Hence, universal verifiability is achieved.

In an actual election, there may be invalid votes as well; then *N* should be taken as the number of valid votes. Also considering the voters who prefer to use a blank vote on purpose, an extra prime number can be associated with a blank vote in the SCP.

4.3 Randomization of RSA for Homomorphic Tallying

The absence of random parameters in the RSA algorithm is a disadvantage, especially when the size of the message space is small so that the probability of collisions among ciphertexts increases. This is a serious drawback within the context of e-voting, where the number of messages to be encrypted is equal to the number of candidates and each ballot is encrypted with the same public key, i.e., the key of the tallying authority. Because of the lack of randomization, a vote *m* given to a specific candidate always yields the same ciphertext under the given public key; hence, encryption cannot provide secrecy.

In different applications, RSA algorithm is usually randomized by adding random padding bits to the plaintext. However, this doesn't work for homomorphic tallying; because, a randomization that would change the unique prime factorization in the vote product *VP* is not allowable since it would destroy the main idea. Below, we detail this problem and present four randomization solutions for homomorphic RSA tallying in sections 4.3.1 to 4.3.4. We use the abbreviations *VP* and *EVP* respectively, for the vote product and the encrypted vote product defined in Section 4.2; and call their randomized forms $VP_{ran}=R\times VP$ and $EVP_{ran}=E_{RSA}\{R\times VP\}$, where R is the overall randomization factor. For each method described below, a vote receipt is in the form $E_{RSA}\{rp_i\}$, where $p_i \in$ SCP and r is a random number. Receipts also contain a unique identification number for the systematic search on the PBB.

- **Why padding bits do not work for multiplicative homomorphic tallying?**

As mentioned above, a common approach to randomize the RSA algorithm is message padding, i.e., adding some random bits to the message *m* so that it becomes another text *m*' with unknown random bits at known locations. After decrypting

$c=E(m')$, it is easy to erase the random bits at known locations of m' to obtain the actual message m back. In homomorphic tallying, this creates a serious problem because the encrypted vote product $EVP=c_1\,c_2\,...\,c_N$ corresponding to a total of N voters is to be decrypted jointly.

Without randomization, decryption of $EVP=c_1\,c_2\,...\,c_N$ would yield the vote product $VP = (p_1)^{v1}\ (p_2)^{v2}\ \dots\ (p_C)^{v_C}$, where C is the total number of candidates. Unique factorization of VP would result in the separate vote counts $v_1, v_2, \dots, v_C$. If padded random messages, $q_1, q_2, \dots, q_C$, were used instead of the prime numbers $p_1, p_2, \dots, p_C$, it would be impossible to find the correct vote counts $v_1, v_2, \dots, v_C$ from the new vote product $VP'=(q_1)^{v1}(q_2)^{v2}\dots(q_C)^{v_C}$, since each of the random messages q_i would be equal to the product of an unknown number of unknown prime numbers. So, when the new vote product $VP'=(q_1)^{v1}(q_2)^{v2}\dots(q_C)^{v_C}$ is factorized into some prime numbers as $VP'=(p_1)^{w1}\ (p_2)^{w2}\ \dots\ (p_B)^{w_B}$, the number of prime numbers in the factorization, $B\neq C$, and the powers of these prime numbers, $w_i\neq v_i$, would be different from those in the original VP, and the correct vote counts $v_1, v_2, \dots, v_C$ would be lost.

4.3.1 Random Shift of the Prime Numbers

Our first solution for RSA randomization consists of the random shift of each prime number represented in bits on the ballot, so that the prime factorization of VP is multiplied by a random power of 2. The main point here is not to assign the prime number 2 to any candidate, and keep it for the randomization of the RSA encryption. The randomization is as follows:

i) The prime number 2 is not assigned to any candidate in SCP and used for randomization.

ii) A vote for the i^{th} candidate is associated with $(2^{s_{ij}}p_i)$, where p_i is a prime number greater than 2; and s_{ij} is randomly chosen in the interval $\{1, \dots, M\}$, where M is the number of extra locations that can be allocated for randomization on the ballot.

Receipts are in the form $E_{RSA}\{2^{s_{ij}}p_i\}$. So, once the randomized encrypted vote product EVP_{ran} is decrypted, the new vote product

$$VP_{ran} = 2^{S}(p_1)^{v_1}(p_2)^{v_2}\ldots(p_C)^{v_C}, \tag{4.18}$$

is obtained. Since $VP_{ran} = R \times VP$, the randomization factor is $R = 2^{S}$. In the vote product $VP_{ran} = 2^{S}(p_1)^{v_1}(p_2)^{v_2}\ldots(p_C)^{v_C}$, each vote is counted as $(2^{s_{ij}}\,p_i)$ where j=1, …, v_i, so the share of candidate i enters into VP_{ran} as $2^{(s_{i1}+\ldots+s_{iv_i})}\,(p_i)^{v_i}$, and hence the power of 2 in VP_{ran} is

$$S = (s_{11} + s_{12} + \ldots + s_{1v_1}) + (s_{21} + \ldots + s_{2v_2}) + \ldots + (s_{C1} + \ldots + s_{Cv_C}).$$

Number of terms in S is equal to the number of all voters, N. It should be noticed that in order to find $2^{s_{ij}}p_i$, provided that it is less than the RSA modulus n, one shifts the binary representation of p_i, simply s_{ij} times (say to the left) and add s_{ij} many 0's (say to the right). Randomization load is cancelled by dividing VP_{ran} to R. Finally, unique factorization of VP results in the separate vote counts $v_1, v_2, \ldots, v_C$.

4.3.2 Randomization Using the "Full Set of Candidate-Primes, SCP" as in Rivest's ThreeBallot Method

The ThreeBallot method proposed in [Rivest-2006], [Rivest-Smith-2007] is intended as a voting scheme that doesn't use any cryptography. Every voter in ThreeBallot-voting fills three ballots as follows: For the candidate he chooses, he votes twice in two different ballots; then for all the remaining candidates that he doesn't choose, he votes once in a single ballot. Therefore, in the final vote count of each candidate, there occurs a superfluous quantity of N votes, that is equal to the number of voters. This extra amount is then subtracted from the final count to obtain the actual vote counts. Among the three ballots, voter chooses one and takes home as his receipt; but the receipt doesn't prove anything to other people because it merely contains a collection of candidates, each having a single vote. As the receipt is formed by only one of the three ballots, vote coercion is not possible; because two similar votes on two ballots are needed in order to prove how a voter has voted.

For the use of a similar concept with homomorphic RSA tallying in a C-candidate election, the software of the voting device prepares three ballots for each voter by distributing the partial products chosen from the "Set of Candidate-Primes", SCP = $\{p_1, p_2, \ldots, p_C\}$, to three ballots randomly, so that each prime appears once in one of the three partial products. Voter's decision is, say candidate j. Since the prime number p_j associated with this choice appears in one of the three ballots, there remain two ballots that do not contain p_j. After voter's decision, the partial product on one of these ballots is multiplied by an extra p_j to indicate the vote. Hence; the software guarantees that although the product of prime numbers $p_1p_2p_3\ldots(p_j)^2\ldots p_C$ on three ballots contains two p_j's, yet none of the partial products on a single ballot repeats any prime number twice. After the RSA encryption of each ballot, software chooses one of them arbitrarily as the receipt and prints together with voter's ID. The randomness of this step is crucial in providing anonymity; because vote coercion is completely prevented as a result of this randomness, since the receipt may or may not contain the actual vote p_j. Notice that such an effect could not be obtained by using two ballots instead of three.

The software then encrypts the contents of the three ballots and multiplies them with other votes to find the EVP_{ran} that is transmitted to the tallying office at the end of the election. Every interested party has access to all receipts on the PBB, and can verify the election results; hence, universal verifiability of the scheme is preserved. The two ballots that are not given to the voter are published on the PBB with no ID number and at random places. Otherwise, anonymity of the vote would be lost and voter verifiability could not be satisfied.

For example, if the encrypted ballots are partitioned as $E_{RSA}(p_1p_2)$, $E_{RSA}(p_3\ldots p_j)$, $E_{RSA}(p_j\ldots p_C)$, their product $E_{RSA}(p_1p_2)E_{RSA}(p_3\ldots p_j)E_{RSA}(p_j\ldots p_C)$ during homomorphic tallying is decrypted as $p_1p_2p_3\ldots(p_j)^2\ldots p_C$. So for each vote given to the candidate j, there is an extra burden of whole product $p_1p_2p_3\ldots p_C$ of SCP elements, multiplied by the actual vote p_j. Since there are N voters, the overall load of this method is a randomization factor $R = (p_1p_2p_3\ldots p_C)^N$ that multiplies VP. Tallying office then cancels this overall randomization load by dividing EVP_{ran} to $E_{RSA}\{R\}$ before decryption, or by dividing VP_{ran} to R after decryption. Finally VP and the corresponding vote counts $v_1, v_2,\ldots, v_C$ are computed.

Although the use of more than three ballots per voter would also serve the purpose of obscuring the actual vote; in order to keep the PBB load at minimum, three ballots per voter seems sufficient. The above discussion encourages some new ideas for the randomization of homomorphic RSA tallying that require the announcement of single receipt per voter as explained below.

4.3.3 Randomization with "Uniformly Chosen Subsets of the Set of Candidate-Primes, SCP"

In an election with C candidates, the use of the randomization concept of the previous section generates a vote product with an extra burden of $p_1p_2p_3\ldots p_C$ multiplied by the actual vote p_j, for each vote given to the candidate j. Uniform insertion of C prime numbers into three ballots does not alter the final vote count, since they can be cancelled deterministically after the election. Whenever there are N voters, this randomness load is reflected to the overall vote product as $(p_1p_2p_3\ldots p_C)^N$.

We now propose to distribute randomization uniformly into successive voters of a voting booth, rather than into three ballots of the same voter. If the associated candidate primes corresponding to C>2 successive voters of the voting booth are multiplied by single primes p_1 or p_2, or p_C that are chosen randomly but uniformly from the "Set of Candidate-Primes", SCP = $\{p_1, p_2, p_3, \ldots, p_C\}$, the randomization load is reflected to the overall vote product as $(p_1p_2p_3\ldots p_C)^{N/C}$, which is a known deterministic number. Then, the randomized encrypted vote product $EVP_{ran}=E_{RSA}\{(p_1p_2p_3\ldots p_C)^{N/C}(p_1)^{v1}(p_2)^{v2}\ldots(p_C)^{v_C}\}$ is decrypted to find the randomized vote product $VP_{ran}=(p_1p_2p_3\ldots p_C)^{N/C}(p_1)^{v1}(p_2)^{v2}\ldots(p_C)^{v_C}$. All parameters of the randomization factor $(p_1p_2p_3\ldots p_n)^{N/C}$ are known, so by dividing VP_{ran} to $(p_1p_2p_3\ldots p_C)^{N/C}$ one obtains the actual vote product $VP{=}(p_1)^{v1}\ (p_2)^{v2}\ \ldots\ (p_C)^{v_C}$. Vote counts $\{v_1,\ v_2,\ldots,\ v_C\}$ are then found by successively dividing VP to each of the associated primes $\{p_1, p_2,\ldots, p_C\}$ until no more division is possible.

Receipts given to the voters are in the form $E_{RSA}(p_i\ p_j)$, where $i{\neq}j$, p_j is the actual vote, p_i is the randomizing prime or vice versa. So, the receipts of voters who vote for different candidates can be the same (for instance $E_{RSA}(p_1p_3)$, can be the receipt of the voter who votes for candidate 1 or candidate 3). Receipts also contain a unique identification number for the systematic search on the PBB.

Randomization level can be further increased by dividing the set of C prime numbers into less than C randomization subsets; say into L sets, where $2 \leq L \leq .C$. These subsets are chosen randomly, with the restriction that prime factors of the randomization factor r does not coincide with the actual vote. Hence, the receipt is in the form $E_{RSA}(r\ p_j)$, where r and p_j have no common factors.

For example, if C=6 and L=2, and two subsets are chosen as $\{r_1{=}p_1p_2p_3p_4p_5,\ r_2{=}\ p_6\}$, then r_1 can be used to randomize the vote p_6 only, whereas r_2 can be used for any vote different from p_6. Choosing L=2 again, two other randomization factors, say $\{r_1{=}p_1p_2p_3p_4,\ r_2{=}p_5p_6\}$ can randomize, the votes p_5 and p_4 of two voters respectively. For L=3, the 3 factors $\{r_1{=}p_1p_2,\ r_2{=}p_3p_4p_5,\ r_3{=}p_6\}$ can randomize 3 votes, or any other permutation like $\{r{=}p_6,\ r{=}p_2,\ r{=}p_3,\ r{=}p_1,\ r{=}p_5p_4\}$ can be utilized for the randomization of 5 votes; each time using all elements of the set of candidate-primes uniformly over 2, 3 or C voters. The important point is never to use a randomization set that coincides with the actual vote; therefore, each prime number can occur in the receipt only once. (Note that if L=1 case were included in the description of the present method, then the set of candidate-primes could totally be taken as a randomization factor as in the method of Section 4.3.2. However, the corresponding single receipt would not be allowable; because, if the vote was used, say for candidate 1, the receipt $E_{RSA}\{(p_1)^2p_2p_3p_4p_5\ p_6\}$ would repeat p_1; and this is the reason for excluding L=1 in the description of the present method.)

All possible subsets of the "Set of Candidate-Primes", SCP = $\{p_1, p_2, p_3, \ldots, p_C\}$ can be considered as a randomization set that will be used to multiply the actual vote, with the constraint that all primes are used with uniform frequency and a randomization set always excludes the unique prime number p_j associated with the vote. The overall encrypted randomized vote product EVP_{ran} is given by $E_{RSA}\{R\times (p_1)^{v1} (p_2)^{v2}\ldots(p_C)^{vC}\}$, where $R = (p_1p_2p_3\ldots p_C)^A$ and A is the cumulative exponent that shows how many times the set $\{p_1, p_2, p_3, \ldots, p_C\}$ is used by all voting booths. At the end of the election period, each of the V voting booths (say A_k for the k'th one) sends its randomization exponent to the tallying office, which computes the cumulative randomization exponent A by summing up all the exponents $A_1, A_2, \ldots, A_V$ coming from voting booths. Tallying office then decrypts EVP_{ran} to find VP_{ran} and multiplies it by R^{-1}. In Section 4.4.3, we show that blind cancellation of R is also possible, only knowing N. Finally VP and the corresponding vote counts $v_1, v_2,\ldots, v_C$ are computed.

4.3.4 Randomization with an Arbitrary Number Followed by Its Inverse

Our fourth randomization suggestion is different from the above three methods; because it randomizes the vote of the j'th voter by multiplying it with a random number r_j, and then cancels this randomization factor by multiplying one of the successive votes by r_j^{-1} (modulo the RSA modulus n). Overall contribution to the final vote product is then $r_j.r_j^{-1}$=1 (modulo n). So, in an election with N voters, instead of the cancellation of N randomization factors together at the end of the election period, mutually exclusive small groups of votes continuously cancel the random factors of each other, during the election.

As an example, say individual votes of the successive 4 voters are p_2, p_C, p_1 and p_3, i.e., they are used for candidates 2, C, 1 and 3 respectively. To randomize these votes, one can use the random factors r_1, r_2 and r_3, to obtain the encrypted votes $E_{RSA}(r_1p_2)$, $E_{RSA}(r_2p_C)$, $E_{RSA}(r_3p_1)$ and finally, the fourth vote that cancels the previous three randomization factors is $E_{RSA}(r_1^{-1}r_2^{-1}r_3^{-1}p_3)$. Decrypted vote product of these 4 votes is $p_2p_Cp_1p_3$, since $r_1r_2r_3$ cancels $r_1^{-1}r_2^{-1}r_3^{-1}$ (modulo n).

To increase the resistance of the method against coercion, the size (that is chosen as 4 in the previous example) of the group of votes that cancel the randomization factor of each other is also randomized by the software of the voting machines. Mutually exclusive voter groups of size 2,3,4,…,K chosen arbitrarily cancel the randomization factors within the group, where K can be chosen as say, 20, 30, …100, depending on the application size. The software should certainly be open to the investigation and check of any interested party before and after the elections.

Overall system becomes more easily post-auditable with regard to randomness cancellation (i.e., the correctness of cancellation can be more directly verified), if the random numbers r_j and r_j^{-1} used in the above method are chosen such that none of their prime factors coincide with the original set of candidate-primes SCP = $\{p_1, p_2,\ldots, p_C\}$.

In Table 4.1, we summarize all randomization methods proposed above for homomorphic RSA tallying.

Table 4.1 Summary of randomization methods for homomorphic RSA tallying, assuming N voters and C candidates with associated prime numbers p_i and vote counts v_i.

Randomization Type	**Randomized Vote Product**
1) Random Shift	$VP_{ran}=2^S(p_1)^{v1}\,(p_2)^{v2}\ldots(p_C)^{v_C}$, where $S=\sum_{i=1}^{N}s_i$
2) Full SCP with three ballots per vote	$VP_{ran}=(p_1p_2p_3\ldots p_C)^N(p_1)^{v1}\,(p_2)^{v2}\ldots(p_C)^{v_C}$
3) Uniformly chosen subsets of SCP	$VP_{ran}=(p_1p_2p_3\ldots p_C)^A(p_1)^{v1}(p_2)^{v2}\ldots(p_C)^{v_C}$, where $(N/C)\le A\le(N/2)$.
4) Arbitrary numbers and their inverses	$VP_{ran}=(r_1r_2r_3\ldots r_{N/2})(r_1r_2r_3\ldots r_{N/2})^{-1}\,(p_1)^{v1}(p_2)^{v2}\ldots(p_C)^{v_C}$

4.4 Some Practical Considerations about Proposed Randomization Methods

We discuss the size of the overall vote product, number of operations per vote, maximum possible size of the voter set for a given RSA modulus n and blind cancellation of the overall randomization factor R. We then compare, criticize and comment on the feasibility of proposed randomization schemes in practice.

4.4.1 Randomization Load and Voter Set Size for RSA Tallying

We compute the size of the randomized vote product VP_{ran} and the average number of operations per vote for each case, and summarize in Table 4.2.

1) Random Shift of Prime Numbers

As mentioned in Section 4.3.1, after the randomization of each vote by random shifting, the new vote product is given by (4.18) as $VP_{ran}= 2^S (p_1)^{v1} (p_2)^{v2}\ldots(p_C)^{v_C}$. Each randomized vote in (4.18) is of the form $2^s p_i$, where the random number s is chosen in the interval $\{1,\ldots, M\}$ and the prime number p_i is represented by at most J bits. Since $2^s p_i < 2^M 2^J=2^{M+J}$, each vote is upper bounded by 2^{M+J} and can be represented by $(M+J)$ bits. Assuming that there are N voters in a given election region, the product of N such votes is less than $(2^{M+J})^N=2^{(M+J)N}$. Hence, the final randomized vote product, $VP_{ran}= 2^S(p_1)^{v1} (p_2)^{v2} \ldots (p_C)^{v_C}$, can be represented at most by $(M+J)N$ bits, MN bits for the randomization shift factor S and JN bits for the overall product of the candidate-prime numbers.

2) Full Set of Candidate-Primes, SCP

For the use of homomorphic RSA tallying together with the full set of candidate-primes, SCP, and three ballots per vote, exponents in the randomized vote product $VP_{ran}= (p_1p_2p_3\ldots p_C)^N(p_1)^{v_1} (p_2)^{v_2}\ldots (p_C)^{v_C}$ sum up to $CN + v_1 +\ldots+ v_C = (C+1)N$.

Hence, assuming that at most J bits are needed to represent each prime, the randomized vote product occupies a maximum size of $(C+1)JN$ bits.

3) Uniformly Chosen Subsets of the Set of Candidate-Primes, SCP

This case is similar to the previous case, except that the exponents in the randomized vote product $VP_{ran} = (p_1p_2p_3\ldots p_C)^A\,(p_1)^{v_1}\,(p_2)^{v_2}\ldots\,(p_C)^{v_C}$ sum up to $CA+v_1+\ldots+v_C = CA+N$, where A is between N/C and $N/2$. So, the number of prime numbers in VP_{ran} is between $2N$ and $((C/2)+1)N$. Assuming that at most J bits are needed to represent each prime, VP_{ran} occupies a maximum storage location between $2JN$ and $((C/2)+1)JN$ bits.

4) Arbitrary Numbers and Their Inverses

This method makes cancellation of randomization during the election, so VP_{ran} occupies at most JN bits and it doesn't require any extra location for randomization. We summarize the results of this section in Table 4.2.

Table 4.2 Maximum size of the randomized vote product and number of operations required for homomorphic RSA tallying with different randomization methods, assuming C candidates, N voters, at most J bits for each prime number and M bits for random shift.

Randomization Type	**Max. Size of the Randomized Vote Product**	**Number of Operations per Vote**
1) Random Shift	$MN+JN$ bits	1 RSA encryption of the vote $2^s p_j$
2) Full SCP with three ballots per vote	$(C+1)JN$ bits	1 RSA encryption of the vote $(p_1p_2p_3\ldots p_C)\,p_j$
3) Uniformly chosen subsets of SCP	$2JN$ to $((C/2)+1)JN$ bits	1 RSA encryption of the vote, say $(p_1)p_j$ or $(p_1p_3)p_j$ or $(p_2p_3\,p_5)p_j$
4) Arbitrary numbers and their inverses	JN bits	1 RSA encryption of the vote, rp_j or $r^{-1}p_j$

For all of the above cases, average number of operations per vote is equal to a single

RSA encryption. It can be kept small by choosing the number of 1's in the public RSA exponent e of the tallying authority small. There is no such constraint on the secret key d, since the decryption of the encrypted vote product is a single operation that is not repeated. After performing the single decryption operation that yields the vote product, vote counts of candidates are found by N successive division operations on the vote product.

Notice that, the parameter JN appears in each term of the second column of Table 4.2; because it corresponds to the size of the vote product VP before randomization. The additional storage required for randomization is equal to MN bits for randomization by shift, CJN bits for randomization with full SCP, and between JN and $(C/2)JN$ for randomization with uniformly chosen subsets of SCP. There is no storage need for randomization parameters in the last method; since the randomization terms are continuously cancelled, the randomized vote product VP_{ran} occupies at most JN bits, like VP.

In Appendix A, we show that the overall randomization can be cancelled at the final tallying, and there is no need to adjust the RSA modulus size according to the extra randomness load reflected on the randomized vote product. Provided that $VP<n$, decryption of $E(VP)$ equals VP after the cancellation of randomization. Hence, the only restriction is to keep the bit length of VP less than the bit length of the RSA modulus; so $JN < (\log_2 n)$. Therefore, the voter set size is upper bounded by $N < (\log_2 n)/J$ for all randomization methods discussed above. Since the primes in the SCP are J bits or less, there is also a margin of security in the bound $N < (\log_2 n)/J$.

As an example, let us choose C=14 as in the 2007 Turkish elections. The first 14 prime numbers excluding 2 are {3,5,7,11,13,17,19,23,29,31,37,41,43,47} (see Appendix B for the first 250 prime numbers). Number of bits required to represent these primes are at most 6 bits. So, with J=6 bits and $\log_2 n = 2^{14} = 16{,}384$ bit-RSA, homomorphic tallying of 2730 votes is possible (that can be considered as the size of 7-8 ballot boxes). If $2^{24} = 16{,}777{,}216$ bit-RSA were used instead, then the size of the voter set could be as large as 2,796,202; therefore, suitable for homomorphic tallying of a large election region like İstanbul-1 (where 2,130,644 votes were used in 2007, July 22 elections). In Table 4.3, we show the maximum voter set size N that we find using the bound $N < (\log_2 n)/J$ for different number of candidates.

Table 4.3 Suggested maximum size N of the voter set with cancelled randomization load for different modulus values, assuming at most J=4,5and 6 bits for each prime number.

Randomization Type	Maximum Bit Size for Each Prime, J	Maximum Voter Set Size N for RSA with a Modulus Size ($\log_2 n$)				
		2^{11}-bit	2^{12}-bit	2^{14}-bit	2^{18}-bit	2^{24}-bit
Random Shift, Full SCP with Three Ballots, Uniform Subsets of SCP or Arbitrary Primes and Their Inverses	4	512	1024	4096	65,536	4,194,304
	5	409	819	3276	52,428	3,355,444
	6	341	682	2730	43,690	2,796,202

For $\log_2 n = 2^{24}$, one needs two prime numbers, p and q, of size 2^{23} bits. Generation of prime numbers up to n using "Sieve of Eratosthenes" algorithm has a complexity of $O(n.\log n.\log\log n)$ [Pritchard-1987]. The time complexity of this algorithm in RAM machine model is given as $O(n(\log\log n))$ [Atkin-Bernstein-2004]. So, generation of all primes up to 2^{23}-bit numbers seems to have a huge time complexity of $O(2^{2^{23}}(23))$. However, the problem of finding some of the large primes is more accessible than finding all primes, as the results of the "Great Internet Mersenne Prime Search (GIMPS)" with Cooperative Computing Award of \$100,000 demonstrate. The largest known prime number announced (by July 23, 2010) at contest's web page [GIMPS-2010], contains 43,112,609 (i.e., more than 2^{25}) bits. The product of two such primes would yield an RSA modulus of more than 86 million bits; which would be suitable for a 2-candidate election with SCP={2,3} and 86/2=43 million voters, or a 14-candidate election with SCP={2,3,5,7,11,13, 17, 19,23,29,31,37,41,43} and 86/6 ≈ 14 million voters.

4.4.2 Detection of the Overall Randomization Load for Cancellation

For all the randomization methods described in Section 4.3, the randomized vote product VP_{ran} can be expressed as the product of two terms, $VP_{ran}=R\times VP$, where R refers to the overall randomization parameter that should be identified at the end of the election period. Tallying office then cancels R and finds $VP=R^{-1}\times VP_{ran}$ (mod n) correctly if $VP<n$ (related preliminaries from number theory and explanation of why there is no need to keep $R<n$ are given in Appendix A). We re-tabulate VP_{ran} and R values for the randomization by "shift", "full SCP", "uniformly chosen subsets of SCP" in Table 4.4. The fourth method, i.e., the "arbitrary numbers and their inverses", is also included to demonstrate that the random numbers used for half of the votes cancel the other half, and the overall product $R_n=(r_1r_2\ldots r_{N/2})(r_1r_2\ldots r_{N/2})^{-1}$ becomes equal to 1 at the end of the election.

Table 4.4 Randomized vote product VP_{ran} and its random part R, assuming N voters and C candidates with associated prime numbers p_i and vote counts v_i.

Randomization Method	**Randomized Vote Product, VP_{ran}**	**Randomization Parameter R in VP_{ran}**
1) Random Shift	$VP_{ran}=2^S(p_1)^{v_1}(p_2)^{v_2}\ldots(p_C)^{v_C}=R\times VP$	$R=2^S$, where $S=\sum_{i=1}^{N}s_i$
2) Full SCP with three ballots per vote	$VP_{ran}=(p_1p_2p_3\ldots p_C)^N\ VP=R\times VP$	$R=(p_1p_2p_3\ldots p_C)^N$
3) Uniformly chosen subsets of SCP	$VP_{ran}=(p_1p_2p_3\ldots p_C)^A\ VP=R\times VP$, where $(N/C)\leq A\leq(N/2)$.	$R=(p_1p_2p_3\ldots p_C)^A$, where $(N/C)\leq A\leq(N/2)$.
4) Arbitrary numbers and their inverses	$VP_{ran}=(r_1r_2r_3\ldots r_{N/2})(r_1r_2r_3\ldots r_{N/2})^{-1}\ VP$	$R=(r_1r_2\ldots r_{N/2})(r_1r_2\ldots r_{N/2})^{-1}=1$ where r_j's are arbitrary.

In Table 4.4, the first randomization parameter is $R=2^S$. For its cancellation at the tallying office, the open-auditable software of each of the V voting booths should keep the sum of individual shift parameters s_{ij} and send their sum S_k to the tallying office at the end of the election, together with all receipts. Then the talliers compute the overall exponent S by summing up all the exponents S_1, S_2, …, S_V coming from all voting booths. A vital advantage, brought by the cancellation of randomization load, is the wide range of bits that can be allocated for randomization by shift. For instance, with 16,384-bit RSA, one can even allocate M=16,378 bits for the random shifts, if primes associated with candidates are represented by at most J_{max}=6 bits.

In the second method, using "full SCP with three ballots per vote", $R=(p_1p_2p_3\ldots p_C)^N$ is already known at the end of the election period, after N voters use their votes. In the third case, uniformly distributed subsets of SCP create a randomization load, $R=(p_1p_2p_3\ldots p_C)^A$, where A is the cumulative exponent that shows how many times the SCP is used for randomization. At the end of the election period, each of the V voting booths (say A_k for the k'th one) sends its randomization exponent to the tallying office, which computes the cumulative randomization exponent A by summing up all the exponents A_1, A_2, …, A_V coming from voting booths.

Since uniform choice of primes requires the use of SCP for an integer number of times, each exponent A_k is an integer and the software of a voting booth must complete its last set $\{p_1, p_2, \ldots, p_C\}$ at the end of the election period, if all primes of the last set are not used. For this purpose, the software finally transmits an empty vote randomized with the remaining primes of the last SCP.

4.4.3 Blind Cancellation of Randomization Load

In the previous sections, we have proposed four effective randomization methods for homomorphic RSA tallying. For easier reference we call these methods:

M1 (Method 1): Random shifts,
M2 (Method 2): Full SCP with three ballots,
M3 (Method 3): Uniform subsets of SCP,
M4 (Method 4): Arbirary number and its inverse.

For the first three methods, and for M4 with cancellation group size chosen as N (i.e., cancellation at the tallying office), overall randomization factor R in the randomized vote product $VP_{ran} = R \times (p_1)^{v_1} \ldots (p_C)^{v_C}$ should be cancelled to find the vote counts v_i.

- For M2, $R = (p_1 p_2 p_3 \ldots p_C)^N$ is already known; hence it can be easily cancelled.
- For M3, $R = (p_1 p_2 p_3 \ldots p_C)^A$, where $(N/C) \leq A \leq N/2$. So, calling $B = p_1 p_2 p_3 \ldots p_C$, *blind cancellation of R* can be done by the following algorithm:

```
0) $VP_{est} = VP_{ran}$
1) Multiply $VP_{est}$ by $B^{-1}$, call it $VP_{est} = B^{-1} \times VP_{est} = (p_1)^{w1} \ldots (p_C)^{wC}$.
2) Find $w_1, \ldots, w_C$. If the sum $w_1 + \ldots + w_C$ is not equal to $N$, go to step 1.
3) Stop. Vote counts are $v_i = w_i$ for $i=1,\ldots,C$.
```

- For M1, an algorithm similar to the above one is used with B=2.

```
0) $VP_{est} = VP_{ran}$
1) Multiply $VP_{est}$ by $B^{-1}$, call it $VP_{est} = B^{-1} \times VP_{est} = 2^{w0}(p_1)^{w1} \ldots (p_C)^{wC}$.
2) Find $w_1, \ldots, w_C$. If the sum $w_1 + \ldots + w_C$ is not equal to $N$, go to step 1.
3) Stop. Vote counts are $v_i = w_i$ for $i=1,\ldots,C$ and $w_0=0$.
```

- For M4, blind cancellation is not possible if all voting booths do not send their randomization factor products to the tallying office.

4.4.4 Comparison and Critics of the Proposed RSA Randomization Methods and Our Suggestions for Implementation

In this section, we try to compare and criticize our randomization methods in two groups, M2-M3 that use SCP elements and M1-M4 that do not; so that we can make a choice among them to shape our final randomization suggestion for an implementation that employs homomorphic RSA tallying. The above mentioned randomizations differ in two major aspects:

1) Last method, M4, cancels the randomization terms of successive votes continuously during the election, within some non-intersecting groups of votes; whereas the first three methods M1, M2, M3 make this cancellation at the end of the election period, for once. Randomization parameter R brings no extra constraint on the size of the modulus n, as shown in Appendix A.
2) Randomization parameters are chosen from outside the SCP in the first method M1; inside the SCP in the second and third methods, M2 and M3; and they can be inside or outside the SCP in the fourth method M4.

M2 and M3: Using SCP for randomization, together with the concept of three ballots as in M2, brings an important advantage: Since nobody can know whether the receipt, chosen by the software out of three ballots, does or does not contain the actual vote; anonymity of the vote is achieved against unlimited computational power. Disadvantages of M2, with respect to M3, are the additional computations needed for the preparation of the two extra ballots; and the amplified storage need on the PBB resulting from the storage of three ballots per voter. Alternatively, M3 has the disadvantage (against an adversary that computes encryptions of all possible 2^C combinations of SCP elements) of revealing the set of candidates, A, and its complement A', such that SCP=$A \cup A'$, and the used vote is within the set A.

The number of different receipt types (of the form $E\{r_i.p_j\}$, where r_i and p_j are both greater than one) for M2 and M3 are limited to $(2^C - C - 1)$ for an election with C candidates. Since voters may prefer to see many receipt types on the PBB that are different from theirs to feel more confident about the privacy of their votes, limited number of receipt types in M2 and M3 may be a disadvantage for the elections with small number of candidates.

M1 and M4: In the fourth randomization method M4, which uses continuous cancellation of randomization during the election, the tallying office's job of "randomness cancellation" is distributed in time and space to individual voting booths. Hence, the software running on voting machines now becomes responsible for the cancellation of randomization factors instead of transmitting them to the tallying office. A particularly interesting application that makes M4 directly post-auditable (with regard to randomization cancellation), is to choose the random numbers r_j and r_j^{-1} such that all their prime factors are outside the SCP. Then, cancellation of randomization becomes easily controllable after decryption of the overall vote product, since any non-cancelled random factor would be distinguishable from the candidate-primes used for voting.

The same advantage of being directly post-auditable also exists in the first method M1 that multiples the vote with random powers of the prime number 2, which are also outside the SCP. However, for M1, it is possible to argue that, no matter how large the number of extra locations M for random shifts is chosen, an adversary who has high computational power can compute all possible encryptions of $(2^s p_i)$ for all $s=1,\ldots,M$ and for all primes p_i assigned to $i=1,\ldots,C$ candidates to create a database of all possible encrypted votes. Then the adversary may attempt vote coercion by comparing the receipts announced on the PBB against his database.

On the other hand; M4 can be criticized as having a disadvantage in preserving anonymity against an adversary with infinite power as follows: Adversary picks up all possible combinations of k receipts from the PBB. If the randomization cancellation of M4 is being done within voter groups of size up to K, then $2 \leq k \leq K$, and with N receipts on the PBB, there are $\binom{N}{2}+\binom{N}{3}+\ldots+\binom{N}{K}$ combinations to be tried, which can be really hard for large values of N and K. However, adversary starts with $k=2$, and multiplies all possible $\begin{pmatrix} N \\ 2 \end{pmatrix}$ receipt pairs on the PBB, to arrive at possible cancellations of some random parameters. Remembering that each receipt is in the form $\mathrm{E}\{r_i.p_j\}$; whenever a cancellation occurs between receipts, adversary arrives at $\mathrm{E}\{r_i.p_j\}\times\mathrm{E}\{r_i^{-1}p_l\}=\mathrm{E}\{r_i.r_i^{-1}p_j.p_l\}=\mathrm{E}\{p_j.p_l\}$. Since he has the computational power to prepare the encrypted forms of all possible (at most) C^2 vote product pairs $\mathrm{E}\{p_j.p_l\}$ (where C is the number of candidates), adversary is able to find a collision that will suffice him to understand the votes of the two receipts that match with $\mathrm{E}\{p_j.p_l\}$. He then proceeds with $k=3,\ldots,K$ and tries to find collisions in receipt sets of size $3,\ldots,K$, respectively; each time by comparing $\begin{pmatrix} N \\ k \end{pmatrix}$ receipt products of k voters chosen from the PBB, with previously prepared (at most) C^k encrypted vote products.

This is why we think that the cancellation of the overall randomization at the end of the election period is a stronger way of preserving anonymity than using M4. On the other hand; if some of the random numbers r_i of M4 are chosen from the SCP, the system can be protected against the attack described above because randomization numbers can then be easily mixed up with the primes used for the actual votes.

Our Suggestion for Randomization of RSA Tallying: The joint use of M2, i.e., “Full SCP with three ballots”, and M3, i.e., “Uniform subsets of SCP”, offers a practical system with drastically increased anonymity for the randomization of RSA tallying. In this joint implementation that we call M2/M3, any one of the two randomizations can be used for each voter, randomly during the election. The open-audit software is responsible for the random choice between M2 or M3, and this choice is invisible on the voter side, who takes a receipt from the machine without knowing which one of the two methods is used in his case.

In order to increase the number of receipts $(2^C - C - 1)$ of M2/M3, each M2 or M3 receipt of the form $E\{p_i...p_j\}$ can be further randomized by M1, i.e., by "Random shifts", to obtain $E\{2^{s_k}p_i...p_j\}$. The resulting method that we call M1+(M2/M3) is our primary randomization suggestion for homomorphic RSA implementation. Because, this joint implementation combines all advantages of the three mentioned methods:

1) Because of using M2, whose receipts may or may not contain the actual vote, anonymity against unlimited computational power is achieved. A corrupt party does not have any chance of extracting information from an individual receipt, even if it has the large power to decrypt the receipt; either by computing all possible encrypted votes for all possible sets of prime number combinations and for all possible randomizations, or by stealing the secret key of the tallying authority.
2) Because of using M3, the number of receipts on the PBB is less than $3N$. If M2 is utilized D% of the time and M3 is used in the remaining $(100-D)$%, the PBB load N is multiplied by $a = [3D + (100-D)] \times 0.01 = 1 + 0.02D$ that is less than 3, whenever $D<100$.
3) Because of using M1, the number of receipt types is multiplied by $(\log_2 n-J)$. With an RSA modulus n, possible receipt types can be as large as $(\log_2 n-J)(2^C - C - 1)$, for a C-candidate election that uses J bits to represent the maximum element of SCP.
4) Blind cancellation of the overall randomization parameter R is possible knowing N, even if none of the voting booths send their randomization factors.

Whenever the number of receipts $(2^C - C - 1)$ is much larger than the number of voters N, it is possible to use only M2/M3 by dropping M1. In Section 5.3, we describe the details of such an implementation proposal for C=18, $(2^C - C - 1)=$ 262,125 and N=3000.

A secondary randomization option can be the multiplication of each vote p_j by arbitrary numbers r as in M4; but leaving the cancellation to the end of the election period as in the other methods. Each voting booth stores the product of randomization factors used by all voters, and transmits the product to the tallying office; so that the overall product *R* can be found at the tallying office and cancelled by R^{-1} by using a single inversion operation. The advantage of a single inversion as opposed to continuous inversions during the election is two-fold: *i*) System becomes resistant to the attack (on M4) described above and *ii*) the inversion operation is not repeated many times during the day. Disadvantage of this method with respect to our primary preference M1+(M2/M3) is the lack of the blind cancellation property.

4.5 Conclusion

Utilizing the concept of *Vote Product* (*VP*) instead of the *Vote Sum* (*VS*) of additive homomorphic tallying, we have described prime factorization of *VP* and employed it for multiplicative RSA tallying.

Since the usual RSA randomization by padding does not work for homomorphic tallying, we have proposed four new methods of RSA randomization: 1) Random shifts, 2) Full set of candidate-primes (SCP) with three ballots, 3) Uniform subsets of SCP, 4) Arbitrary number and its inverse. We have compared these methods, discussed their advantage and disadvantages and proposed a joint randomization using the second and third ones when the number of candidates (that is the size of SCP) is large enough. Otherwise we have suggested multiplication by randomization parameters chosen from outside the SCP as well, in order to add different receipt types on the PBB so that the receipt set is enlarged. We have also shown in Appendix A, why the size of the overall randomization parameter does not bring any restriction to the modulus bit size $\log_2 n$ of the RSA algorithm.

The work presented in Chapter 4 has been the core of the two submitted papers [Yücel-Baykal-2010-b] to ICEG 2010, "6th International Conference on E-Government,", and [Yücel-Baykal-2010-c] to IEEE Transactions on Information Forensics and Security.

CHAPTER 5

COMPARISON WITH OTHER HOMOMORPHIC SCHEMES AND AN IMPLEMENTATION PROPOSAL

We first compare homomorphic RSA tallying with other additive and multiplicative homomorphic tallying algorithms in Section 5.1. After the presentation of our simulation results in Section 5.2, we give the details of an implementation proposal for Turkish Parliamentary Elections that uses homomorphic RSA tallying with proper randomization, in Section 5.3.

5.1 Multiplicative Homomorphic RSA Tallying versus Other Homomorphic Tallying Methods

Multiplicative homomorphic tallying was first proposed in [Peng-Aditya-Boyd-Dawson-Lee-2004], where El Gamal algorithm is employed as the encryption method. Peng et al claim that when the number of candidates is small, their scheme is "more efficient than the additive homomorphic e-voting schemes and more efficient than other voting schemes". Main public key algorithms used for additive homomorphic tallying are Exponential El Gamal, first proposed in [Cramer-Gennaro-Schoenmakers-1997] and Paillier, first proposed in [Damgârd-Jurik-2001]. Below, we compare these algorithms with the multiplicative homomorphic ones, El Gamal and RSA.

RSA algorithm for homomorphic tallying is first proposed in our work; most probably, the main obstacle for other researchers being the randomization problem associated with homomorphic RSA tallying, which is elegantly solved by our randomization proposals explained in Section 4.3 and detailed in Section 4.4.

In the comparison of algorithms versus required number of operations, we use the previously given encryption and decryption equations (4.1) to (4.4). We summarize these equations below for easy reference to Table 5.1, where we compare the four algorithms with respect to their efficiency in homomorphic tallying.

RSA Encryption: $c = E(m) = m^e \pmod n$ (5.1)

Decryption: $m = D(c) = c^d \pmod n$

pk is (n, e), where e and $\Phi(n)$ are co-prime, $1<e<\Phi(n)$; sk is d, $ed = 1 \pmod{\Phi(n)}$.

El Gamal Encryption: $c = E(m) = (\alpha, \beta) = (g^r, m\, y^r) \pmod p$ (5.2)

Decryption: $m = D(c) = [\beta / \alpha^x] \pmod p$

pk is $y = g^x \pmod p$; sk is x, selected by the user randomly such that $x \in Z_q^*$, q is a large prime factor of $p-1$, g is the known generator of a q-order subgroup of Z_p^*.

Exp. El Gamal Encryption: $c = E(m) = (\alpha, \beta) = (g^r, g^m y^r) \pmod p$ (5.3)

Decryption: $m = D(c) = [\log_g(\beta / \alpha^x)] \pmod p$

pk is $y = g^x \pmod p$; sk is x, selected by the user.

Paillier Encryption: $c = E(m) = g^m r^n \pmod{n^2}$ for a random $r \in Z_{n^2}^*$ (5.4)

Decryption: $m = D(c) = [L(c^\lambda \pmod{n^2}) / L(g^\lambda \pmod{n^2})] \pmod n$, $L(u)=(u-1)/n$.

pk is (n, g), $n = pq$, order of $g \in Z_{n^2}^*$ is a multiple of n; sk is $\lambda = \operatorname{lcm}(p-1, q-1)$.

Since all public key encryption schemes work in finite multiplicative groups; they use arithmetic modulo some very large integer. For the prime numbers p and q, RSA modulus is $n=pq$, Paillier modulus is $n^2=p^2q^2$, but El Gamal (or Exponential El Gamal) modulus is simply p. RSA and El Gamal algorithms are considered to have approximately the same security if they use the same-size moduli. Therefore, in the following comparison, we assume that they use moduli of the same length. With this assumption, a single multiplication in RSA or El Gamal have equal complexity. On the other hand, Paillier encryption of equivalent security employs the same *n* as RSA, but it uses (mod n^2) operations instead of (mod *n*); hence, Paillier multiplication is considered to be harder than El Gamal or RSA multiplications. In Table 5.1, we compare various properties of the mentioned homomorphic tallying algorithms.

Table 5.1 Comparison of four public key algorithms suitable for homomorphic tallying

Algorithm		**RSA** (**with M1, M2, M3**)	**El Gamal**	**Exp. El Gamal**	**Paillier**
Homomorphism		Multiplicative	Multiplicative	Additive	Additive
Number of Encryption Operations	Exponentiation	1	2	3	2
	Multiplication	1 (for random factor *r*)	1	1	1
Number of Decryption Operations	Discrete Logarithm	-	-	1	-
	Exponentiation	1	1	1	2
	Inversion	1 (for random factor *r*)	1	1	1
	Multiplication	1 (for random factor *r*)	1	1	1
Distributed Key Generation		Efficient	Efficient	Efficient	Highly inefficient
Randomization Power		Infinite	High	High	VeryHigh
Required Modulus Size versus the Number of Voters *N*		O(*N*)	O(*N*)	O(log*N*)	O(log*N*)

Among the encryption and decryption operations ranked in Table 5.1 in the order of decreasing difficulty; the most time consuming one is the discrete logarithm operation that appears in the Exponential El Gamal algorithm only. The plaintext m of the El Gamal algorithm (5.2), is changed to g^m in Exponential El Gamal (5.3). Although the rest is the same as the El Gamal algorithm; one needs to compute an extra exponent in the encryption and then take the discrete logarithm of g^m in the final decryption of the Exponential El Gamal algorithm, which is a very difficult problem that complicates the implementation.

Exponentiation is the second important operation of Table 5.1, but its difficulty is much less than that of the discrete logarithm operation. Since an exponentiation consists of many successive multiplications, a multiplication operation is not worth counting as compared to an exponentiation. Inversion operation takes longer time than multiplication but it can also be performed much more rapidly than exponentiation. We now continue by the explanation and interpretation of each row of Table 5.1.

1) **Encryption:** RSA encryption (5.1) is computationally more efficient as compared to other encryptions, because El Gamal (5.2) requires two, Exponential El Gamal (5.3) needs three and Paillier uses two exponentiations per encryption, whereas RSA requires only one. El Gamal and Paillier algorithms employ their extra exponentiation for providing randomness. Our three RSA methods, M1, M2 and M3 achieve randomization with a single multiplication; and M4 needs $2k$-2 multiplications and 1 inversion for a cancellation group of k votes.

2) **Decryption:** The least efficient one among the four algorithms is Exponential El Gamal because of its need for taking discrete logarithm ($\log_g$ of g^m to find m). It is followed by Paillier that requires two exponentiations (mod n^2).

   El Gamal decryption performs an exponentiation α^x, an inversion $(\alpha^x)^{-1}$, and then a multiplication by β. Alternatively, RSA decryption seems to need a single exponentiation c^d to get the message m. However, for a fair comparison, we should consider the cancellation of the randomization parameter R, where our first three randomization schemes extract VP from the message $m=R\times VP$; so an inversion (R^{-1}) and a multiplication ($VP=m\times R^{-1}$) is added to the computations required for randomized RSA. Hence RSA decryption becomes completely equivalent to El Gamal decryption.

Our fourth randomization method M4 does not need the mentioned inversion and multiplication operations in the final decryption, because of its continuous cancellation of randomization. However, these two operations do not have noteworthy contribution to complexity as much as an exponentiation operation. Even if they did, since decryption would be performed only once in homomorphic tallying, such slight differences would not be important while choosing a cryptosystem. On the other hand; M4 has the disadvantage of needing the inversions r^{-1} during the election period, for some votes that will cancel the randomization parameters of other votes.

3) **Ease of distributed key generation:** An important concern of homomorphic e-voting is to provide all security measures for keeping the secret key of the tallying office strongly protected. Since the encrypted vote product is to be decrypted once, and using the secret key of the tallying authority; the system should provide extreme care on generating, preserving and storing the secret key. No officer alone is given to hold this responsibility; instead, a group of trusted officers share different and non-overlapping segments of the secret key and decryption can only be performed when all officers come together to combine the separate parts of information. This also necessitates the ability of the related public key algorithms to generate the secret key in a distributed manner, so that no single person can access the entire secret key and is able to perform the final decryption alone. Efficient distributed key generation algorithms exist for RSA [Frankel-MacKenzie-Yung-1998], [Fouque-Stern-2001], as well as for El Gamal [Gennaro-Jarecki-Krawczyk-Rabin-1999]. Paillier algorithm has a modulus

$n^2=p^2q^2$, more complicated than the other algorithms, which makes the distributed key generation harder.

4) **Power of randomization to provide anonymity:** Considering (5.2), (5.3), where the public key of the tallying office is $y = g^x \pmod{p}$, and y, g and p are publicly known, there is a chance of an adversary with infinite computational power to compute very large number of receipts $c = E(m) = (g^r,\ m\,y^r) \pmod{p}$, for many possible values of the randomization factor $r<p$, with a small probability of collision. Same argument is also valid for Paillier randomization given by (5.4), but with a smaller probability since $r \in Z^*_{n^2}$. Our hybrid RSA randomization proposals that employ the concept of the "full SCP in one of three ballots" reduce this chance to zero and provide anonymity against infinite computational power.

5) **Required Modulus Size:** One of the main differences between additive and multiplicative homomorphic tallying algorithms lies in their essence: Additive homomorphic tallying employs the overall vote sum *VS*, whereas multiplicative homomorphic tallying employs the vote product *VP* for the same purpose. In order that the *VS* or *VP* can be recovered correctly in modulo operations, they should not exceed the modulus of the algorithm. The sum of *N* votes *VS*, occupies a bit length proportional to $\log_2 N$, whereas the product of *N* votes, *VP*, has a bit length proportional to *N*. Therefore, additive homomorphic tallying has the advantage of requiring much smaller modulus size of order $O(\log N)$ as compared to the modulus size of order $O(N)$ needed for multiplicative homomorphic tallying. On the other hand, the security of the implementation increases as the modulus size increases.

In summary, the main problems with additive homomorphic algorithms are as follows: Exponential El Gamal necessitates finding the vote sum *VS* by taking the discrete logarithm ($\log_g$) of g^{VS}, which is a very difficult problem. Paillier lacks an efficient distributed key generation algorithm, whereas the existence of such algorithms for distributed key generation is an important criterion. On the other hand, they both have the advantage of requiring much smaller moduli for a given number of voters. Nevertheless; there is a continuous and fruitful interest on the development of efficient algorithms for the generation of large primes and it is more practical to perform homomorphic tallying, by dividing the election regions into sub-regions of smaller voter set size (see Section 5.3), where each sub-region uses a single PBB for announcing the receipts but leaves the decryption of the randomized vote product EVP_{ran} to the main tallying office of the election region.

As for the comparison among the multiplicative homomorphic algorithms, RSA tallying is more efficiently implementable than El Gamal tallying, mainly because of *i*) its smaller encryption complexity, that is equal to half of the El Gamal encryption and *ii*) its randomization power to provide infinite anonymity. The choice of the set of candidate primes, SCP, considering the quadratic residue or non-residue elements of the group, is more complicated [Peng-Aditya-Boyd-Dawson-Lee-2004], as compared to our simple choice of the smallest primes for RSA. However, we don't think that this is a significant difference since SCP is generated once, before the elections.

One of the important differences between El Gamal and RSA algorithms is that, El Gamal security depends on the hardness of the discrete logarithm problem, whereas RSA security is a consequence of the difficulty of factorization of numbers into their large prime factors. The asymptotic running time of the best discrete log algorithm is approximately the same as that of the best factoring algorithm [Schneier-1996]. Therefore, it requires about as much effort to solve the discrete log problem modulo a 256-bit prime, as to factor a 256-bit RSA modulus. Historically, an algorithmic advance in one of these problems was then applied to the other. For the future, nobody knows which one of these problems will provide more security; therefore, it is much rational to develop e-voting schemes for various public key algorithms, whose security depend on both problems. So, RSA tallying with our randomization proposals should be considered as an additional and efficient option in multiplicative homomorphic tallying, whose security depends on the hardness of the prime factorization rather than the discrete logarithm problem.

5.2 Simulation Results

5.2.1 RSA Modulus Generation and RSA Tallying

We have simulated homomorphic RSA tallying for a possible set of election parameters, by using the Magma library developed at the University of Sydney, which can deal with unlimited-precision integers. The main parameter that determines the system constraints is the bit size $\log_2 n$ of the RSA modulus n; because, the number of voters, N, that can be handled by our proposal, is strictly upper bounded by $N<(\log_2 n)/J_{AV}$, where J_{AV} is the average number of bits assigned per prime p_i, associated with each candidate. However, to use this bound as $N<(\log_2 n)/J$, where J is the maximum number of bits used per prime p_j, $j=1,\ldots, C$ is safer, as we have done in our simulations. Recent applications that use RSA encryption employ modulus bit sizes like $\log_2 n$ =1024 or 2048 bits; nevertheless, there are also cases that employ 4096 bits, 8192 bits or even to $16384=2^{14}$ bits. In Table 5.2, we show the average CPU time that we have spent for generating the RSA primes p and q to obtain a modulus $n=pq$ of bit size $\log_2 n$.

Table 5.2 Average time required for generating RSA primes p, q and the modulus n, using MAGMA library and a 1,83 GHz CPU.

Bit size $\log_2 n$ of the RSA modulus n	128	256	512	1024	2048
Average generation time	0.05 seconds	0.15-0.17 seconds	0.8-1.3 seconds	16-33 seconds	15-16 minutes
Relative duration wrt 128 bit case	1	~3	~16-26	~320-660	~18000-19200
Relative duration wrt 1024 bits	~0.002	~0.006	~0.04	1	~40

For generating a 4096-bit RSA modulus, we have utilized a faster CPU having a clock rate of 3.16 GHz. Generation of the 2048-bit modulus took ~10 minutes on this system and 4096-bit modulus was generated in 17372 seconds ≈ 4 hours and 50 minutes; which is approximately 30 times of the duration required for a 2048-bit modulus on the same 3.16 GHz CPU.

We have considered three cases in the simulations: I) No randomization of votes, II) Randomization by shift, III) Randomization by a single prime. In order to simulate multiplicative RSA tallying with a modulus bit size of $\log_2 n$ and C candidates, each one being represented by a prime number p_j, j=1,…, C, of maximum bit size J, we have followed our simulation algorithm below:

1) Calculate the primes p, q, $n=pq$ and the corresponding public and secret keys of the tallying office.
2) Compute the number of voters, N, that is upper bounded by $(\log_2 n)/J$, such that the vote product VP does not exceed n. Initialize the vote number as $i=0$.
3) Choose a "vote", as a prime number p_i randomly from the set of C primes that can be represented by at most J bits.
4) Multiply the prime number chosen at step 3 by one of the randomization factors r_i given below, depending on the chosen case: I) $r_i=1$; II) $r_i=2^{s_i}$, $s_i<M$; III) $r_i=p_j, j\neq i$.
5) Encrypt the randomized vote r_ip_i to find the $E(r_ip_i)$.
6) Let the vote number be $i=i+1$. Go to step 3 if $i<N$.
7) Find the product EVP_{ran} of all encrypted votes $E(r_ip_i)$ for i=1,…, N.
8) Find the product R of all randomization terms r_i for i=1,…, N, to keep the record of the cumulative randomization.
9) Decrypt EVP_{ran} to find VP_{ran}.
10) Cancel the randomization by performing modular division $VP = VP_{ran}\times R^{-1}$ $= VP_{ran} / R \pmod n$ before the final decryption and find VP.
11) Divide VP to all primes p_j for j=1, …, C as many times as possible (with no remainder) to evaluate the vote counts v_j for j=1, …, C.
12) Print the vote counts v_j for j=1, …, C, and stop.

In Table 5.3, we summarize our simulation results, where the last column demonstrates the time spent for N encryptions plus final decryption for tallying.

Table 5.3 Average time required for performing all computations of the election with homomorphic RSA tallying with randomization, using MAGMA library and a 1,83 GHz CPU.

Bit size of RSA modulus	**Bit size *J* per vote**	**Randomization bits & case**	***N*, Number of votes**	***VP*, vote product**	**CPU time (sec) for generating *p*, *q*, *e*, *d***	**CPU time (sec) for the election**
16	2	*M*=4, II	8	$2^5 3^3$	0.027	0.036
16	2	*M*=4, II	8	$2^3 3^5$	0.033	0.052
20	2	*M*=4, II	10	$2^4 3^6$	0.071	0.078
20	2	*M*=18, II	10	$2^5 3^5$	0.062	0.047
24	2	*M*=18, II	12	$2^6 3^6$	0.039	0.203
24	2	*M*=18, II	12	$2^5 3^7$	0.051	0.484
24	3	*M*=20, II	8	$2^2 3^3 5^1 7^2$	0.047	0.312
24	3	*M*=10, II	8	$2^3 3^2 5^2 7^1$	0.058	0.983
24	3	*M*=10, II	8	$2^3 3^1 5^2 7^2$	0.072	1.529
32	3	I	10	$2^1 3^1 5^4 7^4$	0.073	0.114
40	2	I	20	$2^9 3^{11}$	0.064	0.032
128	2	I	64	$2^{31} 3^{33}$	0.094	0.047
128	2	I	64	$2^{35} 3^{29}$	0.045	0.109
256	2	I	128	$2^{66} 3^{62}$	0.141	0.046
256	2	I	128	$2^{59} 3^{69}$	0.172	0.171
512	2	I	256	$2^{136} 3^{120}$	1.045	0.047
512	2	I	256	$2^{124} 3^{132}$	1.321	0.110
512	4	I	128	$2^{16} 3^{10} 5^{24} 7^{25} 11^{29} 13^{24}$	1.045	0.031
512	4	I	128	$2^{10} 3^{17} 5^{31} 7^{22} 11^{22} 13^{26}$	0.950	0.063
1024	2	I	512	$2^{254} 3^{258}$	18.562	0.281
1024	2	I	512	$2^{272} 3^{240}$	33.103	0.265
2048	2	I	1024	$2^{516} 3^{508}$	587.854	0.485
2048	2	I	1024	$2^{522} 3^{502}$	587.854	0.578
2048	4	I	512	$2^{56} 3^{49} 5^{94} 7^{113} 11^{92} 13^{108}$	587.854	0.391
1024	3	3 bits, III	170	$2^{30} 3^{32} 5^{58} 7^{50}$	16.8	0.078
2048	3	3 bits, III	340	$2^{80} 3^{72} 5^{101} 7^{87}$	587.854	1.235
4096	3	3 bits, III	682	$2^{114} 3^{93} 5^{215} 7^{260}$	17372.350	3.354
8192	3	3 bits, III	1364	$2^{392} 3^{343} 5^{338} 7^{291}$	486416.23	7.265
30000	4	*M*=2000, II	7500	$3^{847} 5^{1658} 7^{1680} 11^{1668} 13^{1647}$	0.27 (*p* and *q* generation excluded)	2434 (1047 for decrypt)

We give the program that cancels the randomization parameter in Appendix C.

The last row of Table 5.3 uses 15,000-bit primes p and q, generated at a government office in 3 hours. Finally, we have simulated the last part of an election (step 11) with C=6 candidates and N=1,000,000 voters; to measure the time required for the final step to extract the vote counts v_1, …, v_C from the vote product VP= $(p_1)^{v1}\ldots(p_C)^{v_C}$, where 6 prime numbers associated with candidates are less than J=4 bits. Division of VP 1,000,000 times to these primes took 6691 seconds ≈ 1hour and 50 minutes on a 1,83 MHz CPU.

5.2.2 Measurement of CPU Times for Exponentiation, Inversion and Multiplication

In order to observe how the theoretical comparison of the modular operations mentioned in Section 5.1 is supported experimentally, we have performed the modular multiplication, inversion and exponentiation operations employed by RSA, El Gamal and Paillier encryptions using the Magma library. We have measured the CPU times corresponding to 100,000 operations and noticed that in a multiplicative group with given modulus, multiplication and inversion times are negligible with respect to the time required for an exponentiation (Appendix E). For instance, with a 1024-bit modulus, product of two 640-bit numbers takes only 1/2483'th of the time required for their exponentiation, and inversion of a 640-bit number can be performed in 1/423'th (see the last row of Table E.2 in Appendix E) of an exponentiation time. Therefore, we concentrate on the exponentiation, and observe that the required operation time depends linearly on the exponent size of g^r. For a specific modulus size, exponentiation time is approximately doubled, whenever the bit size of the exponent r is doubled.

To find the dependence of the operation speed on the bit size of the modulus, we have measured the CPU times versus different values of the modulus and shown some results (summarized from Appendix E) in Table 5.4. It is observed that for the same size exponents, the operation time is approximately tripled whenever the modulus size is doubled (also see Table E.3). As the average size of random elements picked up in a multiplicative group is doubled with doubling the modulus size as well; for the exponentiation of arbitrarily picked elements, spent CPU time is almost six times more, if the modulus size of the multiplicative group is doubled.

Table 5.4 Average CPU times corresponding to 100,000 modular operations with 512-bit and 1024-bit moduli versus the size of the group elements, using MAGMA library and a 1,83 GHz CPU.

Modulus Size	**Multiplicative Group Elements *g* and *r* of Size**	**Multiplication** $g \times r$	**Inversion** g^{-1}	**Exponentiation** g^{r}
512	20 bits each	0.063	0.406	25.397
1024		0.078	0.592	79.857 3
512	40 bits each	0.093	0.64	6 59.53
1024		0.078	0.874	197.154
512	80 bits each	0.094	1.061	129.263
1024		0.125	1.56	6 431.295 2
512	160 bits each	0.156	1.919	277.838
1024		0.156	2.402	926.163
512	320 bits each	0.608	3.806	664.471
1024		0.312	4.899	6 2079.50 3
512	640 bits each	2.356	6.302	1427.315 2
1024		1.857	10.733	4611.40

As for the comparison of different public key algorithms, one needs a fair basis; such as equal security level. So, in order to have "prime factorization" and "discrete logarithm" problems of equal hardness level, we assume (*n* of RSA) = (*p* of El Gamal) = (*n* of Paillier) for the comparison in Table 5.5. We only consider the most time consuming operations: exponentiation and discrete logarithm, the latter one being much more difficult than the former. Multiplication by 6 in the last column of

Table 5.5 is the result of doubling the modulus bit size, since Paillier uses log n^2 operations.

Table 5.5 Rough ratio of average CPU times found for encryption and decryption, by using operations of MAGMA library for public key algorithms at similar security level.

Algorithm	RSA	El Gamal	Exponential El Gamal	Paillier
Encryption	1	2	3	2×6
Decryption	1	1	1+DL time>>1	2×6

Ratios in Table 5.5 are rough and deduced from the values given in Table 5.1, for the number of exponentiations. They can be more detailed by taking into account that the exponentiation m^e of RSA encryption uses a fixed exponent e; but El Gamal and Paillier employ exponentiations like g^r, y^r and r^n, where the random numbers r can be very large, and n is always very large. In Table E.2, one observes that with a 1024-bit modulus and 640-bit g, 640-bit exponentiation g^r takes 4611.4 seconds, whereas 17-bit exponentiation g^e (with e= 65537) can be performed in only 88.7 seconds. Since 88.7/4611.4=0.02, time ratio of RSA encryption in Table 5.5 is more correctly represented by 0.02 instead of 1. Then, by multiplying each element of Table 5.5 with 50, corresponding encryption time ratios become 1: 100 : 150 : 600 for RSA : El Gamal : Exponential El Gamal : Paillier, as shown in Table 5.6.

Table 5.6 Ratio of average CPU times found for encryption and decryption by using MAGMA library for public key algorithms of similar security level, considering an 17-bit public RSA key and randomly picked numbers of size 640 bits, for a 1024-bit modulus.

Algorithm	RSA	El Gamal	Exponential El Gamal	Paillier
Encryption	1	100	150	600
Decryption	50	50	50+DL time>>50	300

Each term in Table 5.6 (except the first entry, 1, that is used for reference) would be approximately halved for randomly picked exponents of 320 bits. In order to observe how close our above prediction is to the actual case, we have simulated both RSA and El Gamal Algorithms for 256-bit, 512-bit and 1024-bit moduli, assuming five candidates with SCP={3,5,7,11,13} and shown the measured CPU times in Table 5.7. Since the maximum number of bits to represent a candidate prime is 4, number of voters is chosen as modulus size divided by 4 in each case.

For a modulus size of 1024 bits; Table 5.7 shows that 256 encryptions are performed in 0.219 sec. for RSA, and in 19.204 sec. (or 36.317 sec. in the 2nd trial) for El Gamal (see Appendix C); corresponding ratio is 88 (and 166 for the 2nd trial), which is of similar order as our expectation (100) in Table 5.6. The main reason for this advantage of RSA is its fixed and small-size exponent *e*. Since the secret key *d* is determined after choosing *e*, such that $ed=1 \pmod{\Phi(n)}$, *d* is not of small size necessarily; hence decryption times of RSA and El Gamal are closer to each other.

Table 5.7 CPU times found for 5-candidate election simulations with homomorphic RSA and El Gamal tallying that have the same modulus size; by using MAGMA library and a 1,83 GHz CPU.

Algorithm	**Modulus Size & Number of Voters**	**Vote Product**	**CPU Times Spent for**			
			Initializa-tion	***N* Encryptions**	**Decryption**	**Final *N* Divisions**
RSA	256 & 64	$3^{8}5^{15}7^{11}11^{18}13^{12}$	0.153	0.078	0	0
El Gamal		$3^{8}5^{12}7^{11}11^{13}13^{20}$	1.255	0.281	0	0
RSA	512 & 128	$3^{13}5^{39}7^{14}11^{30}13^{32}$	1.123	0.048	0.015	0.015
El Gamal		$3^{22}5^{29}7^{20}11^{32}13^{25}$	28.64	1.152	0.047	0.016
RSA	1024 & 256	$3^{19}5^{48}7^{59}11^{64}13^{66}$	42.791	**0.219** (×1)	0.109	0.016
El Gamal		$3^{23}5^{48}7^{67}11^{55}13^{63}$	858.489	**19.204** (×88) **36.317** (×166)	0.094 0.156	0 0.016

5.3 An Implementation Proposal for Turkish Parliamentary Elections

Finally, we describe the implementation details of our proposed e-voting system with homomorphic RSA tallying, as reflected on a real-life example.

Choice of the Election Parameters: Considering Turkish Parliamentary Elections held in 2007 with 14 candidates, we choose an SCP={2,3,5,7,11,13,17,19,23,29,31, 37,41, 43,47,53,59,61} of size C=18, where the last prime numbers can be associated with independent candidates or a blank vote (see Appendix B for the first 250 prime numbers). Each prime number in SCP can be represented by at most J=6 bits. Employing an RSA modulus size of $\log_2 n$=16,384 bits; a voter set size of 16,384/6=2730 has a very large margin. Assigning the smallest primes {2,3,5,7} up to 3 bits to the largest 4 parties (of total vote percentage much greater than 70%), average number of bits per vote is still less than $J_{AV} = 0.7\times3+0.3\times6 = 4.9$ bits. Corresponding voter set size is then N=16,384/4.9=3343.

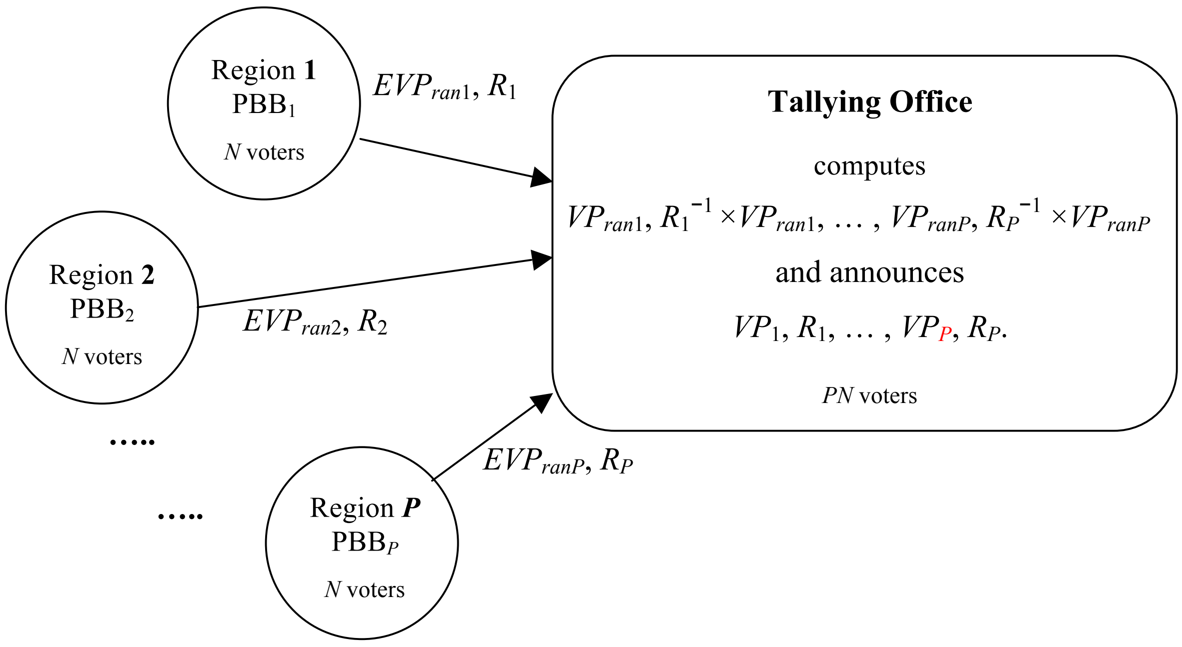

Figure 5.1. E-voting organization of an election region with PN voters and P bulletin boards for homomorphic RSA tallying.

Each election region is divided into P bulletin boards in P sub-regions, as shown in Figure 5.1. In order to find the number of PBB's for each election region, the number of voters in the region is divided by N. If we consider Amasya for example, where 196,021 votes were used in the 2007 elections; the election region could be divided into P=66 sub-regions with N=3000, since PN=198,000 >196,021. For a larger election region like Ankara-1, where 1,281,877 votes were counted in 2007; P=428 with N=3000 would yield PN =1,284,000 >1,281,877.

Voting, Receipts and PBB Announcement: During the election, the voting software at the polls of each sub-region prepares the receipts $E_{\mathrm{RSA}}\{r_i.p_j\}$ for each vote p_j, where the randomization factor r_i is chosen according to the randomization method M2 in 20% and M3 in 80% of the time. The software is open-auditable and these percentages can be checked by any interested party before, during or after the elections. Hence, 600 out of 3000 votes are randomized by M2 and 2400 votes are randomized by M3. The randomization method is chosen by the software randomly and the voter does not know whether it is M2 or M3. The receipt $E_{\mathrm{RSA}}\{r_i.p_j\}$ is given to the voter after voting and it is announced at the PBB together with voter's identity at the end of the election.

The number of different receipt types with C=18 is equal to $(2^C - C - 1)$= 262,125. Since the allowed number of voters in the sub-region is N=3000, there will be $0.02\times3N + 0.08\times N = 1.4\times N$ = 4200 receipts announced on the PBB; 3000 of them with the identities of their voters and an additional number of 1200 receipts with unknown identity, resulting from the use of the second randomization method M2 for 600 voters.

Tallying: All P sub-regions send their encrypted vote product EVP_{ran_k}=E$\{R_k\times VP_k\}$, and the product R_k=$(p_1p_2p_3\ldots p_C)^{Ak}$ of N randomization factors to the tallying office, at the end of the election period. To prevent dealing with very large numbers, modular product R_k.(mod n) or only the exponent A_k of the SCP product $(p_1p_2p_3\ldots p_C)^{Ak}$ is sent. Previously assigned central office talliers come together to form the secret key of the tallying office from its distributed parts. They join their pieces of information to obtain the secret key and decrypt each EVP_{ran_k} to find VP_{ran_k} for k=1,…, P. Randomization is then cancelled by multiplying VP_{ran_k} by R_k^{-1}. Prime factorization of the k'th vote product VP_k yields the vote counts of Region k. Vote counts are announced at each PBB separately; and also for the whole election region.

Blind Cancellation of Randomization: Employment of the randomization methods M2 and M3 allows the cancellation of randomization blindly, even when none of the sub-regions send their SCP exponents A_k or the product R_k (mod n). Knowing that the randomization set is a multiple of $B=p_1p_2p_3\ldots p_C$, overall randomization factor is $R_k=B^{A_k}$, so all one needs is to multiply VP_{ran_k} blindly by B^{-1}, sufficient number of times such that the sum of the vote counts $v_1,v_2,\ldots,v_C$ in the resulting estimated vote product $VP_{est}= (p_1)^{v1}(p_2)^{v2}\ldots(p_C)^{v_C}$ is equal to N. Knowing that M2 is used 20% of the time, the exponent A_k has to be greater than $0.2N$=600. Since M3 picks up the randomization factors by dividing SCP into L sets, where $2\leq L\leq C$, probable values of the exponent A_k are between 600+(2400/18)=733 and 600+(2400/2)=1800. So one can directly start by $VP_{est}=VP_{ran_k}\times B^{-733}$(mod n). For this application, the product of SCP elements is B=($117288381359406970983270_{(decimal)}$=$3744D1E73FE25000_{(hex)}$ = $11011101000100110100011110011100111111111000100101000000000000_{(binary)}$; so B occupies 62 bits.

Voter Verifiability: Each voter is able to see and check his receipt on the PBB of his sub-region. If his receipt is not announced on the PBB, he takes it to the tallying office to make an objection. If the receipt he holds doesn't exist among the recorded receipts, provided that the receipt is not forged, this may be sufficient evidence to repeat some part of the elections or even to cancel them.

Universal Verifiability: Each interested individual can first multiply all 4200 receipts on the k'th PBB to find the product EVP_{ran_k}. Secondly, employing the announced election results, VP_k and the randomization factor R_k, he finds $R_k\times VP_k$ and encrypts it using the public key of the tallying office to form the randomized vote product EVP_{ran_k} once more. If the results of two computations do not match, provided that his computations are provably correct, he has full right for objection. Verifying the correctness of his computations, he can force computations of the tallying office be repeated, corrected or even cancelled.

Anonymity: Receipts given to the voters contain an encrypted product of prime numbers chosen from the SCP, that may or may not contain the unique prime number that is associated with the candidate whom they vote for. Therefore, even an adversary with infinite computational power cannot derive any hint about the used vote. So, the system does not let vote-coercion and vote-selling.

Auditing: Software and all electronic devices used for the election should be auditable before and after the elections, so that the security and reliability of the scheme can be verified. There is no need to use any zero knowledge proofs (ZKP), neither on the voter side, nor on the voting machine side. Voter doesn't need to prove that his vote is valid as in [Benaloh-Yung-1986] because the software does not allow voting incorrectly. Voter also doesn't need to check whether the voting machine encrypts correctly, either by immediate decryption or randomly choosing among hundredths of encrypted versions of the same vote as described in [Benaloh-2006]. Instead, any interested voter can check the open source software (OSS) and audit the electronic machines by using tools as in [Paul-Tanenbaum-2009], where the emphasis is on the audits of OSS by means of the Trusted Platform Module (TPM) that allows the verification of the voting machine in real time, by demonstrating that the machine runs the open source software that it is supposed to run. Neff explains that a small percentage of voters who are interested in making such tests are sufficient to assure a high degree of election accuracy [Neff-2003].

CHAPTER 6

CONCLUSIONS

In this work, we investigate different aspects of e-voting, with special emphasis on auditable, voter/universally verifiable and anonymous schemes. We try to predict possible e-voting technologies of the future and to contribute them.

We start by proposing a modification in the Single Transferable Voting (STV) method so that it can be applied to large scale elections with electoral barriers. We present a case study to demonstrate the effect of preferential voting on the election systems with electoral barriers; by employing the vote counts of 2007 Turkish Parliamentary Elections under four simple and politically unbiased scenarios on the distribution of secondary vote preferences. After the mathematical formulation of the election procedure, we make simulations for the 69 election regions (that have no independent parliament members) by using a combination of the modified STV and d'Hondt methods. Our computations show that, if the voters were given the chance of preferential voting, election results could drastically change. One of our scenarios, in which we assume that the secondary choices of the wasted vote owners are distributed uniformly among the winning parties, is found to yield outcomes very close to the *proportional representation*; i.e., the seat percentages computed by the "modified STV+d'Hondt" method closely match with the actual vote percentages under this scenario.

Modified STV can be properly used with the voter and universally verifiable "Prêt a Voter: All-In-One (PAV-2007)" e-voting scheme [Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007] for the elections with electoral barriers. We revise PAV-2007 by proposing three security enhancing modifications in its ballot construction phase: 1) ballot serial number, 2) digital signature of the first clerk in the mix-net, 3) different random numbers for each row of the ballot.

Since the seminal thesis work [Benaloh-1987] of Benaloh for conducting secret-ballot elections in which the outcome is verifiable by all observers; voter verifiability and universal verifiability by strictly preserving the privacy of votes (i.e., the anonymity) are the main concerns of e-voting schemes. For achieving anonymity, we have focused on homomorphic tallying, which is firstly proposed in [Benaloh-1987], for yes/no votes and distributed government agents (tellers). Benaloh employs the concept of r^{th} residuosity, which depends on the difficulty of finding x in large groups, such that z is an r^{th} residue, i.e., $z=x^r \pmod{n}$. Use of the discrete logarithm problem (difficulty of finding r for given z, x and n, in $z=x^r \pmod{n}$) as the source of homomorphism is first discussed in [Cramer-Franklin-Schoenmakers-Yung-1996] for multi-authority elections. Tallying application for a specific public key algorithm first appears in [Cramer-Gennaro-Schoenmakers-1997], where the El Gamal algorithm is modified as Exponential El Gamal, so that its multiplicative homomorphism is converted into additive homomorphism. Later, additive homomorphic Paillier tallying is considered in [Damgârd-Jurik-2001]. Multiplicative homomorphic tallying is first proposed in [Peng-Aditya-Boyd-Dawson-Lee-2004], where El Gamal algorithm is employed as the encryption method to be used with the concept of prime factorization. Our contribution in this work is the proposal of the RSA algorithm for homomorphic tallying and its randomization specifically for this purpose.

Utilizing the notion of unique prime factorization of the vote product (VP) instead of the vote sum (VS) employed by additive homomorphic tallying; we demonstrate that the RSA algorithm is a promising candidate for multiplicative homomorphic tallying, provided that it can be randomized properly. To indicate a vote, we associate a prime number with each candidate and call the corresponding set "the set of candidate-primes, SCP".

The main obstacle for homomorphic RSA tallying is the lack of randomization associated with RSA that does not spoil the unique factorization of the vote product. We solve this problem by proposing four different types of randomization for RSA tallying. The essence of all these methods consists of multiplication of the prime number indicating the vote, by some random numbers chosen from inside or outside the SCP. One of our randomization methods that utilizes the concept of ThreeBallot proposed in [Rivest-Smith-2007] provides strong anonymity against unlimited computational power.

We also show that the growth of the randomization factor does not bring any extra load to the actual vote product, and it can effectively be cancelled using modular division after the transmission of the overall randomization factor to the tallying office. Hence, the maximum possible size N of the voter set is loosely upper bounded by $(\log_2 n)/J$ for a given RSA modulus n (of length $(\log_2 n)$-bits) and prime numbers p_i assigned to each candidate, of at most J-bit long. A tighter upper bound for N can be taken as $(\log_2 n)/J_{AV}$, where J_{AV} is the average number of bits used for the prime numbers in SCP. In the simulations of homomorphic RSA tallying, we have implemented elections up to N=7500 voters using a 30,000-bit modulus generated in 3 hours; encryption of 7500 votes was completed in 23.12 minutes, decryption, randomness cancellation and extraction of final vote counts for 5 candidates was performed in 17.45 minutes.

As for the comparison among the multiplicative homomorphic algorithms, RSA tallying is more efficiently implementable than El Gamal tallying, mainly because of *i*) its smaller encryption complexity, that is at most equal to half of the El Gamal encryption and *ii*) its randomization power to provide infinite anonymity. Among the main public key algorithms used for additive homomorphic tallying, Exponential El

Gamal decryption has the disadvantage of necessitating a discrete logarithm operation, which is a very difficult problem. Paillier on the other hand, lacks an efficient distributed key generation algorithm, but the existence of such algorithms for distributed key generation is very important for the security of the secret key. Although additive homomorphic algorithms have the advantage of requiring much smaller moduli than multiplicative ones for given number of voters; yet, equal security levels are obtained with similar modulus sizes. Besides, it is much more practical to perform homomorphic tallying, by dividing the election regions into sub-regions of smaller size, where each sub-region of size *N* uses a single PBB for announcing the receipts but leaves the decryption of the randomized vote product EVP_{ran} to the main tallying office of the election region.

Finally, we have suggested an implementation considering Turkish Parliamentary Elections with 18 candidates, using 16384-bit RSA in sub-regions of *N*=3000 voters and combining *P* sub-regions into one of the 85 election regions, so the number of sub-regions *P* varies between 15 (for the smallest election region Bayburt) and 710 (for the largest one: İstanbul-1). We have also explained how blind cancellation of randomization is possible for our randomization methods that use the SCP (Set of Candidate-Primes) elements, without knowing the overall randomization factor.

Further studies should focus on the adaptation of homomorphic tallying to preferential voting and generation of new tools for simplifying the software-auditing process so that it becomes easily accessible and attemptable by ordinary voters.

REFERENCES

[Adida-Neff-2009] B. Adida, C. A. Neff, “Efficient Receipt-Free Ballot Casting Resistant to Covert Channels”, EVT/WOTE’09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, Montreal, August 2009. Available online at http://www.usenix.org/events/evtwote09/tech/.

[Adida-deMarneffe-Pereira-Quisquater-2009] B. Adida, O. de Marneffe, O. Pereira, and J. J. Quisquater, “Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios”, EVT/WOTE’09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, Montreal, August 2009. Available online at http://www.usenix.org/events/evtwote09/tech/.

[Adida-2008] B. Adida, “Helios: web-based open-audit voting”, SS’08, Proc. of the 17th Conference on Security Symposium, pp. 335–348, Berkeley, CA, 2008.

[Adida-Rivest-2006] B. Adida and R. L. Rivest, “Scratch & Vote: Self-Contained Paper-Based Cryptographic Voting”, ACM Workshop on Privacy in the Electronic Society, editors: R. Dingledine and T. Yu, pp. 29-39, October 2006.

[Adida-2006] B. Adida, “Advances in Cryptographic Voting Systems”, Ph.D. Thesis, Massachussets Institute of Technology, August 2006.

[Adida-Neff-2006] B. Adida and C. A. Neff, “Ballot Casting Assurance”, EVT ’06, First Usenix/ACCURATE Electronic Voting Technology Workshop, Vancouver, August 2006. Available online at http://www.usenix.org/ events/evt06/tech/.

[Anderson-2008] M. Anderson, “Open-Source Voting”, IEEE Spectrum, pp. 13-14, October 2008.

[Atkin-Bernstein-2004] A. O. L. Atkin and D. J. Bernstein, "Prime sieves using binary quadratic forms", Mathematics of Computation 73, pp. 1023–1030, 2004.

[Baudron-Fouque-Pointcheval-Stern-Poupard-2001] O. Baudron, P. A. Fouque, D. Pointcheval, J. Stern and G. Poupard, “Practical multi-candidate election system”, PODC 2001, Proc. of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283, Rhode Island, August 2001.

[Benaloh-2006] J. Benaloh, “Simple Verifiable Elections", Proc. EVT’06, Electronic Voting Technology Workshop, Vancouver BC, August 2006. Available online at http://usenix.org/events/evt2006/tech/.

[Benaloh-1987] J. Benaloh, “Verifiable Secret-Ballot Elections", Ph.D. Thesis, Yale University, YALEU/DCS/TR-561, New Haven, CT, December 1987.

[Benaloh-1986] J. Benaloh, “Secret sharing homomorphisms: keeping shares of a secret, secret”, Advances of Cryptology-Crypto’86, LNCS 263, pp. 251-260, 1986.

[Benaloh-Yung-1986] J. Benaloh and M. Yung, “Distributing the power of government to enhance the power of voters”, Proc. ACM Symposium on Principles of Distributed Computing, pp. 52–62, 1986.

[Chaum-Carback-Clark-Essex-Popoveniuc-Rivest-Ryan-Shen-Sherman-2008] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, “Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes”, EVT/WOTE’08, Electronic Voting Technology Workshop, San Jose, CA, July 2008. Available at http://www.usenix.org/events/evt08/tech/.

[Chaum-Essex-Carback-Clark-Popoveniuc-Sherman-Vora-2008] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora, “Scantegrity: End-to-end voter verifiable optical-scan voting”, IEEE Security and Privacy, vol. May/June, 2008.

[Chaum-2006] D. Chaum, “Punchscan”, 2006. Available at http://punchscan.org.

[Chaum-Ryan-Schneider-2005] D. Chaum, P. Y. A. Ryan, and S. Schneider, “A practical, voter-verifiable election scheme,” Proc. ESORICS, editors: S. De Capitani di Vimercati, P. F. Syverson, and D. Gollmann, LNCS 3679, pp. 118-139, 2005.

[Chaum-2004] D. Chaum, “Secret-Ballot Receipts:True Voter-Verifiable Elections”, IEEE Security and Privacy, 2(1): pp. 38–47, Jan/Feb 2004.

[Chaum-1981] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms”, Commun. ACM, 24(2), pp. 84–88, 1981.

[Clarkson-Chong-Myers-2008] M. R. Clarkson, S. Chong, A. C. Myers, “Civitas: Toward a Secure Voting System”, Proc. IEEE Symposium on Security and Privacy, pp. 354-368, May 2008. URL: http://www.cs.cornell.edu/projects/civitas/.

[Cramer-Franklin-Schoenmakers-Yung-1996] R. Cramer, M. Franklin, B. Schoenmakers and M. Yung, “Multiauthority Secret-Ballot Elections with Linear Work”, Eurocrypt’96, editor: Ueli M. Maurer, LNCS 1070, pp. 72–83, 1996.

[Cramer-Gennaro-Schoenmakers-1997] R. Cramer, R. Gennaro, B. Schoenmakers, “A Secure and Optimally Efficient Multi-Authority Election Scheme”, Eurocrypt’97, LNCS 1233, editor: W. Fumy, pp. 481–490, 1997.

[Chasteen-2004] S. Chasteen, “Electronic voting unreliable without receipt, expert says”, February 2004”. Available online at http://news-service.stanford.edu/news/2004/february18/aaas-dillsr-218.html.

[Cohen-Fischer-1985] J. D. Cohen (later Benaloh) and M. J. Fischer, “A robust and verifiable cryptographically secure election scheme”, in FOCS, pp. 372–382, IEEE Computer Society, 1985.

[Damgârd-Jurik-2001] I. Damgârd and M. Jurik, “A generalization, a simplification and some applications of Paillier’s probabilistic public-key system”, Proc. of PKC’2001, Public Key Cryptography, LNCS 1992, pp.119-136, 2001.

[Diffie-Hellman-1976] W. Diffie and M. E. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, volume IT–22, no. 6, pp. 644–654, November 1976.

[Election Data Services-2010] Election Data Services, Voting Equipment Studies. Available at http://www.electiondataservices.com/index.php?content=votequip.

[El Gamal-1985] T. El Gamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, IEEE Transactions on Information Theory, volume IT–31, no. 4, pp. 469-472, July 1985.

[Fisher-Carback-Sherman-2006] K. Fisher, R. Carback and A. Sherman, “Punchscan: Introduction and System Definition of a High-Integrity Election System”, Proc. IAVoSS Workshop On Trustworthy Elections (WOTE’06), editor: P. A. Ryan, Cambridge, June 2006.

[Florida-DoS-2000] Florida Department of State, “Official Results of the November 7, 2000 General Election”, 2000. Available online at http://election.dos.state.fl.us/elections/resultsarchive/SummaryRpt.asp?ElectionDate=11/7/2000&Race=PRE&DATAMODE=.

[Frankel-MacKenzie-Yung-1998] Y. Frankel, P. D. MacKenzie, M. Yung, “Robust Efficient Distributed RSA Key Generation”, Proc. of STOC’98, pp. 663-672, 1998.

[Fouque-Stern-2001] P. A. Fouque, J. Stern, “Fully Distributed Threshold RSA Under Standard Assumptions”, IACR Cryptology ePrint Archive, Report 2001/008, February 2001.

[Gennaro-Jarecki-Krawczyk-Rabin-1999] Gennaro, Jarecki, Krawczyk and Rabin, “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”, Proc. Eurocrypt’99, LNCS 1592, 1999.

[Gerlach-Gasser-2009] J. Gerlach, U. Gasser, “Three Case Studies from Switzerland: E-Voting”, Berkman Center Research Publication, no: 2009-03.1, March 2009.

[GIMPS-2010] George Woltman, 1996, “Great Internet Mersenne Prime Search (GIMPS)”, Electronic Frontier Foundation, http://www.mersenne.org/default.php

[Gumbel-2005] A. Gumbel, “Steal This Vote: Dirty Elections and the Rotten History of Democracy in America”, Nation Books, July 2005.

[Heather-2007] J. Heather, “Implementing STV securely in Prêt à Voter”, Proc. 20th IEEE Computer Security Foundations Symposium (CSF’07), pp. 157–169, Venice, Italy, 2007.

[Hirt-Sako-2000] M. Hirt and K. Sako, “Efficient receipt-free voting based on homomorphic encryption”, Advances in Cryptology – Eurocrypt’2000, Int. Conf. on the Theory and Application of Cryptographic Techniques, editor: B. Preneel, LNCS 1807, pp. 539–556, Belgium, May 2000.

[Jefferson-Rubin-Simons-2007] D. Jefferson, A. D. Rubin, B. Simons, “A Comment on the May 2007 DoD Report on Voting Technologies for UOCAVA Citizens”, June 2007. Available online at http://www.servesecurityreport.org/.

[Jefferson-Rubin-Simons-Wagner-2004] D. Jefferson, A. D. Rubin, B. Simons, D. Wagner, "A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)", 2004. Available at http://www.servesecurityreport.org/.

[Katz-Myers-Ostrovsky-2001] J. Katz, S. Myers and R. Ostrovsky, "Cryptographic counters and applications to electronic voting", Advances in Cryptology – Eurocrypt'2001, Int. Conf. on the Theory and Application of Cryptographic Techniques, editor: B. Pfitzmann, LNCS 2045, pp. 78–92, Austria, May 2001.

[Krimmer-2003] A. Prosser, R. Kofler, R. Krimmer, M. K. Unger, University of Economics and Business Administration Vienna, http://www.e-Voting.at

[Landes-2002] L. Landes, "The Nightmare Scenario Is Here - Computer Voting With No Paper Trail", August 2002. Available online at http://www.ecotalk.org/Dr.RebeccaMercuriComputerVoting.htm.

[Li-Hwang-Lai-2009] C. T. Li, M. S. Hwang and Lai "A Verifiable Electronic Voting Scheme Over the Internet", Proc. Sixth International Conference on Information Technology: New Generations, pp. 449-454, Las Vegas, April 2009.

[Magi-2007] T. Magi, "Practical Security Analysis of E-voting Systems", Tallinn University of Technology, Master Thesis, 2007.

[Mercuri-1992] R. Mercuri, "Voting-machine risks" Communications of ACM, 35(11):138, 1992.

[Neff-2004] C. A. Neff, "Practical High Certainty Intent Verification for Encrypted Votes". Available online at http://votehere.net/vhti/documentation/vsv-2.0.3638.pdf.

[Neff-2003] C. A. Neff, "Election Confidence: A Comparison of Methodologies and Their Relative Effectiveness at Achieving It". Available online at http://www.votehere.net/old/papers/ElectionConfidence.pdf

[Neff-2001] C. A. Neff, “A verifiable secret shuffle and its application to e voting,” in Proc. 8’th ACM Conference on Computers and Communications Security, CSS’01, pp. 116–125, 2001.

[Paillier-1999] P. Paillier, “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,” Advances of Eurocrypt’99, LNCS 1592, pp. 223–238, 1999.

[Paillier-Pointcheval-1999] P. Paillier and D. Pointcheval, “Efficient Public-Key Cryptosystems Provably Secure against Active Adversaries,” Proc. Asiacrypt’99, LNCS 1716, pp. 165–179, 1999.

[Park-Itoh-Kurosawa-1994] C. Park, K. Itoh, and K. Kurosawa, “Efficient anonymous channel and all/nothing election scheme”, Eurocrypt’94, editor: T. Helleseth, LNCS 765, pp. 248–259, 1994.

[Paul-Tanenbaum-2009] N. Paul and A. S. Tanenbaum, "Trustworthy Voting: From Machine to System", Computer (published by IEEE Computer Society), pp. 23-29, May 2009.

[Peng-Aditya-Boyd-Dawson-Lee-2004] K. Peng, R. Aditya, C. Boyd, E. Dawson, B. Lee, “Multiplicative Homomorphic E-Voting”, Indocrypt’2004, editor: A. Canteaut, K. Viswanathan, LNCS 3348, pp. 61-72, 2004.

[PfitzmannB-1994] B. Pfitzmann, “Breaking efficient anonymous channel”, Eurocrypt’94, editor: A. De Santis, LNCS 950, pp. 332–340, May 1994.

[PfitzmannB&A-1990] B. Pfitzmann and A. Pfitzmann, “How to break the direct RSA-implementation of mixes”, Eurocrypt’90, editors: J. J. Quisquater and J. Vandewalle, LNCS 434, pp. 373–381, 1990.

[Pritchard-1987] P. Pritchard, “Linear prime-number sieves: a family tree”, Sci. Comput. Programming 9:1, pp. 17–35, 1987.

[Rivest-2009] R. L. Rivest, "Perspectives on 'End-to-End' Voting Systems", talk at the National Institute of Standards and Technology, October 13, 2009. Available online at http://people.csail.mit.edu/rivest/publications.html.

[Rivest-Smith-2007] R. L. Rivest, W. D. Smith, "Three Voting Protocols: Three Ballot, VAV, and Twin", Proc. EVT'07, Electronic Voting Technology Workshop, Boston, August 2007. Available online at http://www.usenix.org/events/evt07/tech/.

[Rivest-2006] R. L. Rivest, "The ThreeBallot Voting System", Available online at: http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVoting System .pdf

[Rivest-Shamir-Adleman-1978] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Communications of the ACM, 21(2):120–126, February 1978.

[Rubin-2004] A. Rubin, "An Election Day clouded by doubt", October 2004. Available online at http://avirubin.com/vote/op-ed.html.

[Ryan-Teague-2009] P. Y. A. Ryan, V. Teague, "Ballot Permutations in Prêt a Voter", EVT/WOTE'09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, Montreal, August 2009. Available online at http://www.usenix.org/events/evtwote09/tech/.

[Ryan-2008] P. Y. A. Ryan, "Prêt à Voter with Paillier encryption", Journal of Mathematical Modelling of Voting Systems and Elections: Theory and Applications, Special Issue of Mathematical and Computer Modelling, vol. 48, no. 9-10, pp. 1646-1662, November 2008.

[Ryan-2004] P. Y. A. Ryan, "A Variant of the Chaum Voter-Verifiable Scheme," Technical Report of University of Newcastle, CS-TR:864, 2004. Also in Workshop on Issues in the Theory of Security, WITS 2005.

[Ryan-Peacock-2005] P. Ryan and T. Peacock, "Prêt à Voter: a system perspective," Technical Report of University of Newcastle, CS-TR:929, 2005.

[Ryan-Schneider-2006] P. Ryan, and S. Schneider, "Prêt à Voter with re-encryption mixes", Technical Report of University of Newcastle, CS-TR:956, 2006.

[Sako-Kilian-1995] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth", Eurocrypt'95, editors: L. C. Guillou and J. J. Quisquater, LNCS 921, pp. 393–403, 1995.

[Schneier-1996] B. Schneier, Applied Cryptography (2nd ed.), John Wiley & Sons Inc., New York, 1996.

[Shapiro-2004] A. Shapiro, "Absentee Ballots Go Missing in Florida's Broward County", October 2004. Available online at http://www.npr.org/templates/story/story.php?storyId=4131522

[Shubina-Smith-2004] A. M. Shubina and S.W. Smith, "Design and prototype of a coercion resistant, voter verifiable electronic voting system", Proc. Conference on Privacy, Security and Trust, pp. 29–39, October 2004.

[Wack-2006] J. Wack, "Voter Verified Paper Audit Trail Update", March 2006. Available online at http://vote.nist.gov/032906VVPAT-jpw.pdf.

[Walton-2004] M. Walton, "Voting methods under close watch", October 2004. Available online at http://www.cnn. com/2004/TECH/10/26/evote/index.html.

[Xia-Schneider-Heather-Traore-2008] Z. Xia, S. Schneider, J. Heather, J. Traore, "Analysis, Improvement and Simplification of Prêt à Voter with Paillier Encryption", EVT/WOTE'08, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, San Jose, USA, July 2008. Available online at http://www.usenix.org/events/evt08/tech/.

[Xia-Schneider-Heather-Ryan-Lundin-Peel-Howard-2007] Z. Xia, S. Schneider, J. Heather, P. Ryan, D. Lundin, R. Peel and P. Howard, "Prêt a Voter: All-In-One," IAVoSS Workshop On Trustworthy Elections (WOTE 2007) , pp. 47-56, Ottawa, Canada, June 2007.

[Yücel-Baykal-2008] O. Yücel and N. Baykal, "Single Transferable Electronic Voting Protocol for Elections with Barriers," Proc. 3rd Information Security and Cryptology Conference, ISC'08, pp. 237-241, Ankara, Turkey, December 2008.

[Yücel-Baykal-2009] O. Yücel and N. Baykal, "Voter Verifiable and Modified Single Transferable E-Voting for Elections with Electoral Barriers", Proc. ECEG 2009, 9th European Conference on e-Government, pp. 703-711, London, UK, June 2009.

[Yücel-Baykal-2010-a] O. Yücel and N. Baykal, "Prêt à Voter Schemes and Preferential E-Voting with Application to a Case Study", Proc. 4th Information Security and Cryptology Conference, ISC'10, pp. 51-57, Ankara, Turkey, May 2010.

[Yücel-Baykal-2010-b] O. Yücel and N. Baykal, "Homomorphic RSA Tallying and Its Randomization for E-Voting", ICEG'2010, 6th International Conference on e-Government, Cape Town, South Africa, September 2010.

[Yücel-Baykal-2010-c] O. Yücel and N. Baykal, "RSA Algorithm for Multiplicative Homomorphic Tallying in E-Voting", submitted to IEEE Transactions on Information Forensics and Security.

APPENDIX A

Cancellation of Very Large Randomization Terms by Modular Division of Z_n^* Elements

In this Appendix, we explain how cancellation of the multiplicative randomization term R becomes possible when $VP_{ran}= R\times VP$ exceeds the modulus n of the RSA algorithm. We show that if VP is kept less than n, it doesn't matter how large the overall randomization parameter R becomes. Using either $E(R)^{-1}$ or R^{-1} in the multiplicative group of integers modulo n, overall randomization can be effectively cancelled and the encrypted randomized vote product $EVP_{ran}=E(R\times VP)=E(R)\times EVP$ can be uniquely decrypted to give VP. So, the prime factorization of the actual votes is preserved. There is absolutely no problem created by the large size of the randomization factor R, because in the multiplicative group of integers modulo n, R is always replaced by its remainder r that is less than n, whatever the value of R is.

We first give an example to show how one can extract VP from the product $VP_{ran}= R\times VP$ or from its encrypted form, $EVP_{ran}=E(R)\times EVP$ knowing R; and how the actual value of VP can be accurately found provided that $VP<n$, even when $R>>n$.

Example: Remembering that RSA encryption and decryption are given respectively by $E_{RSA}(m)=m^e.(\text{mod}.n)=c$, $D_{RSA}(m)=c^d.(\text{mod}.n)$, where $n=pq$, $\Phi(n)= (p-1)(q-1)$, e and $\Phi(n)$ are co-prime, $1 < e < \Phi(n)$, and $ed =1$ (mod $\Phi(n)$); let the prime factors of n be $p=11$, $q=23$; so $n=pq=253$. Euler's function is $\Phi(n)=(p-1)(q-1)=10\times22=220$ and choosing a public exponent $e=3$, which is co-prime with 220, the public key of the tallying office becomes (n, e)=(253, 3). The secret key is computed as d=147, so that ed =1 (mod $\Phi(n)$).

Assume N=3 voters give their votes to three different candidates represented by the prime numbers 3, 5 and 7. So the vote product is VP=3×5×7=105, with the vote counts v_1=v_2=v_3=1. Let the randomization be done by multiplying with $R=2^{10}$, so $VP_{ran}=2^{10}\times3\times5\times7$=107520. In the ring of integers modulo 253, these numbers are equivalent to

$R = 2^{10}$= 1024 = 12 (mod 253),

$VP_{ran}=2^{10}\times3\times5\times7$=107520 = 248 (mod 253),

$EVP_{ran}=E(2^{10}\times3\times5\times7)$= E(107520)= 107520^3 (mod 253) = 128,

VP=3×5×7=105 = 105 (mod 253), i.e., VP remains the same since it is less than n.

At the tallying office, the product of all randomized votes gives the randomized encrypted vote product EVP_{ran}. The problem is to obtain VP with proper cancellation of randomness parameter, using either R^{-1} (mod n) after decryption of EVP_{ran}; or $E(R)^{-1}$ (mod n) before decryption of EVP_{ran}.

So, the procedure is as follows. Given that EVP_{ran}=128,

I. For "randomness cancellation after decryption", compute:

1) $VP_{ran}= D(EVP_{ran}) = D(128) = 128^{147}$(mod 253)=248,
2) $R^{-1} = 12^{-1}$ (mod 253) = 232,
3) $VP = R^{-1}\times VP_{ran}$(mod 253) = 232 × 248 (mod 253) = 57336 (mod 253)
   = 105 = 3×5×7;

II. For "randomness cancellation before decryption", compute:

1) $E(R) = E(2^{10})= 1024^3$ (mod 253) = 210,
2) $E(R)^{-1} = 210^{-1}$ (mod 253) =100,
3) $EVP = E(R)^{-1}\times EVP_{ran}$ (mod 253) =100 ×239 (mod 253) = 150,
4) $VP = D(EVP) = D(150) =150^{147}$ (mod 253) = 105 = 3×5×7.

In both cases, the true value of $VP = 105 = 3\times5\times7$ and corresponding vote counts $v_1=v_2=v_3=1$, given to three different candidates can be found easily.

So when the overall randomization factor R is known, it can be cancelled in two ways, and the vote product VP is found correctly if it does not exceed n:

I. First decrypt EVP_{ran} to get VP_{ran},. Then cancel the randomization by multiplying VP_{ran} with R^{-1} and obtain VP.

II. First multiply EVP_{ran} by $E(R)^{-1}$, and cancel the randomization. Because of the multiplicative homomorphism of RSA: $EVP_{ran}= E_{RSA}(R\times VP) = E(R)\times E(VP)$; hence, when EVP_{ran} is multiplied by $E(R)^{-1}$, $E(VP)=EVP$ is found. Then the RSA decryption, $D(E(VP)$, yields VP.

No information in the vote product VP is lost unless VP doesn't exceed n, regardless of how large VP_{ran} is and how much it exceeds n. It should be remembered that the procedure defined in this example is applied at the tallying office only once for a collection of N voters and works properly iff N is chosen so that $JN < \log_2 n$, i.e., number of bits in the RSA modulus n. When $JN < \log_2 n$, $VP < n$ is satisfied automatically.

Some of the Related Number Theoretical Concepts

In the multiplicative group Z_n^* of integers modulo $n=pq$, there are $\Phi(n)= (p-1)(q-1)$ $(=pq-p-q+1=(pq-1)-(p-1)-(q-1))$ integers, each one having an inverse modulo n. $\Phi(n)$ is called the Euler's totient function. Notice that not all $(n-1)$ elements of $Z_n=\{1,2,\ldots, n-1\}$ are included in Z_n^* because integers like p, $2p,\ldots, (q-1)p$ and q, $2q, \ldots, (p-1)q$ do not have multiplicative inverses.

Fact A.1: Let A, R, B be three integers in Z (they are also elements of the field of real numbers, $\mathcal{R}$) and a, r, b be their respective remainders, if divided by n, in Z_n^*, i.e., $a=A(\text{mod } n)$, $r=R(\text{mod } n)$ and $b=B(\text{mod } n)$. Then $AR=B$ implies $ar(\text{mod } n)=b$.

Proof: Since a, r, b are the remainders of A, R and B (mod n); one can simply take the modulo n of both sides of $AR=B$ to obtain $ar(\text{mod } n)=b$.

Definition A.1: Let A, R, B, $n \in$ Z (and $\in \mathcal{R}$) and let a, r, b be their respective remainders when divided by n. Also let $AR=B$. The inverse R^{-1} does not exist in Z but it exists in $\mathcal{R}$; hence $A=R^{-1}B$, in $\mathcal{R}$. For all integers r in the multiplicative group Z_n^*, the modular division operation is defined as $a = r^{-1}b = b/r$ (mod n) = $(b+kn)/r$, where k is an integer that makes $(b+kn)$ some integer multiple of r. Because $a \in Z_n^*$, the result of the modular division b/r (mod n) cannot take a non-integer value; so it is interpreted as $a=(b+kn)/r$, for some integer k that yields $a \in Z_n^*$.

Now, we can consider VP, R and VP_{ran} of the equation $VP\times R= VP_{ran}$ respectively as A, R and B mentioned in the equation $AR=B$. Since VP is less than n, its remainder is equal to itself, so $A=a$. If R and B are much larger than n, this creates no problem; because in the multiplicative group Z_n^* of integers modulo $n=pq$, R and B are replaced by their remainders r and b that are always less than n. This is why there is no need to put any constraint on the size of the randomization factor R.

The only constraint on R is that it should be invertible, i.e., r^{-1} should exist. Our choice of the randomization factors as powers of 2, or as products of prime numbers from the SCP (see Table 4.3) guarantees that R is not a power of p or q; therefore it is invertible, i.e., $r=R$ (mod n) is in the multiplicative group Z_n^*.

Inversion operation can be considered to be equivalent to an exponentiation; because in a multiplicative group Z_n^* of size $\Phi(n)$, where Φ is the Euler's totient function; for any r in Z_n^* raised to the exponent $\Phi(n)$, $r^{\Phi(n)}(\text{mod } n)=1$; hence, the inverse of r can be found as, $r^{-1}= r^{\Phi(n)-1}$ (mod n). However, there are also other methods such as the extended Euclidean algorithm that finds the greatest common divisor of two integers, which is used to perform inversion much more rapidly than exponentiation.

APPENDIX B

First 250 Prime Numbers

										First
2	3	5	7	11	13	17	19	23	29	10
31	37	41	43	47	53	59	61	67	71	20
73	79	83	89	97	101	103	107	109	113	30
127	131	137	139	149	151	157	163	167	173	40
179	181	191	193	197	199	211	223	227	229	50
233	239	241	251	257	263	269	271	277	281	60
283	293	307	311	313	317	331	337	347	349	70
353	359	367	373	379	383	389	397	401	409	80
419	421	431	433	439	443	449	457	461	463	90
467	479	487	491	499	503	509	521	523	541	100
547	557	563	569	571	577	587	593	599	601	110
607	613	617	619	631	641	643	647	653	659	120
661	673	677	683	691	701	709	719	727	733	130
739	743	751	757	761	769	773	787	797	809	140
811	821	823	827	829	839	853	857	859	863	150
877	881	883	887	907	911	919	929	937	941	160
947	953	967	971	977	983	991	997	1009	1013	170
1019	1021	1031	1033	1039	1049	1051	1061	1063	1069	180
1087	1091	1093	1097	1103	1109	1117	1123	1129	1151	190
1153	1163	1171	1181	1187	1193	1201	1213	1217	1223	200
1229	1231	1237	1249	1259	1277	1279	1283	1289	1291	210
1297	1301	1303	1307	1319	1321	1327	1361	1367	1373	220
1381	1399	1409	1423	1427	1429	1433	1439	1447	1451	230
1453	1459	1471	1481	1483	1487	1489	1493	1499	1511	240
1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	250

APPENDIX C

Sample MAGMA Programs for the Simulations of Homomorphic RSA and El Gamal Tallying

1. RSA Tallying for *N*=7500 Voters with Randomness Cancellation

```
Magma V2.10-22
n_bit:=30000;
p_bit:= n_bit div 2;
q_bit:= n_bit div 2;
J:=4; //Bit size of candidate's prime number
Sbit:= n_bit-J; // Shift allocation for randomization
N:= n_bit div J; // Number of voters
e:=65537; //  65537
//*******************************************************************
total:= Cputime();
initialization:= Cputime();
//p := RandomPrime(p_bit);
//q := RandomPrime(q_bit);
p:=1424784455771478674680195749784183039000995310681391939746637467
39738605252086633288403025160315837201922951767181717460376133017060
18755259302834337037695122633064020276973221169075967246435781691505
26329749805205642505305662739989161294133642913698882747076306336808
82098943654851748981510185271776865181256425867476389635706816703387
38940474131998140158253615486882900069998694014116260415464955306037
10677176411139790564148565398966662905711129517137541533923051372928
30162409780381163743062956521094770271308035720751890759042154323678
65374147876689380263323680976029311960471986330814428298140345865069
34673827804483198045679257414423589056019146615934958721937458278129
31306130960151132269469927960154748619247002302560952080454829018414
31679827465843568387601461049683147181697245432087332943485555429201
60791921126069019519225993324010727328029655229771672645387602425176
32332598427477533819686710576519702548620678720127374180812945312026
15335663138694274500319169072211290771673218653549755628234511293930
99418071409015476554774311398566238006051729096602963757759498334661
9377694567909049824611948922681232729896768954430590701217887979043
36313055614942121218940526237957673908059675946679284464758698234336
42989954693906877040485881602044191239449529000076619425162074270059
```

45167647477093583764847837394122498708661063894562339755847837688499
31008494074313859366265323377782497603969620780069427698295994284922
66667181052708463705673183104248617552164670117833649646909457471611
67385619901748903368862418683637199295740410211646839870885927705125
14294839994752087068602161247122664660784685076830435586191860607630
44470820633089995080590857041510469797120852035076017559933849291315
92864637045410891753783519606255634378237298560154992695125789560596
40134798846190915680314803855624117850328976792733946814724699701763
22415799810176722693041761338390537724181395833114957385255793750364
41824018260624075453327530044823306808211465539243950750889453123561
59476709792585576224096405790206496464680502090263001788074935201763
87537593429653721319654223603100120133184957184068504121102203922525
84695811598406415722520981134325578691826992265994355100982032954264
79453114847939971118931035980166154434742694688619716043785513888456
00881784279514122645966559330978593213945656201065132377944633434672
74204402677298485655521472785852854410204570812622572719844143440891
273152843365278388868587125003769371189612820861331058030985642503159
82548130502283342058268540531455423775792785709981980111768258506977
05858597112420569347565710724272828173398293209809690223383674501085
61811208257745417214691534182266856622520121662838016959391316492319
68880451017546567300584163415023760324233275910971058032395476153495
16047385500698678315464611139833518360799275792449937523766405203034
14634113447811279475975086906423105636822163420796202576189382801664
76627381809249561625101945507826506664677480407498210543760677098543
90140496682842908792047369328514773847391785688802436458064480428336
53793772853335265132463521820436507217241642031658389336331262200291
03037183175182321781137260926911380862936465520822052707976298309987
45653582312609187619499957810880297281843971100556166355598570042490
38514730149994283186292780432327382363141917782108672613793796037889
78888499469196382473520583183530934972973004465599350584725767203860
66406232076224460154326946213326864757089992325302617552310393132706
44631347959995867701555822607888684342251092640512389308397267301732
37567124800398509518075481961977839603139111094712243458316321265567
87161561844767853770833151407694296841821243238385478606687841382360
00179389630603285398862861733737365115437107810653035934835636914293
05395464884607582385581193712458351910782466397134851682088149800847
12360152791407269404507791246047147244186460616128050281334284726956
63642490652220528036453541198214784918382004236564621871932846098777
11860321271131653332743369305325520185149949402753809505138447620551
16144606723400601532088282880193297753628419260452089260259627177856
72343490583793130604173169948724500595202328926311111052815744507385
20949295608033390759067728092645938208441680561613357915211624244253
77184376955929547544352565846395645681264394783050876803339184364115
36127221281608126133197409073514546281082719322118128548052280941899
62108604812578161642655482130799388118143203025682381741814668037941

36345253575488870030965408282774570859079354837323198692947421906017493484227918941205152554418405627008869606121031711202785111977432461730037106302644101239711587069;

q:=3816084114970909610727700852923765510128469529410855943953405491939650244646025428992040536829544029032081866599313820589996484161381602507668968927763276654961299232229954366864269779219016774907177214421568842219990842137667662684083680088148902208115097668545592332916792007634917542015029403257215155960727714927942425305590725623127605668859791115002185172043311290380480003286812265528003156905667339237139482400237799768873233733618390832671827501176076396156668087465517358104278129610591889536133142565088378463288997134565791660964994460897853620673491405082687713246188338238075882166642955564507051271855772430907392816019511212500602544030036833446046171391514003382729143001725042777499277825242938889474925891788233210044837658078820058539872401558286857306439917406094265698969511473017583486701976053439244427918331783635547335816194313633094577032643682182439556221264697997312351964699223878410586187509492770904369511435698275528950478502534021618779227069944903941240657032010175019818719949313654148368307531301724953786623227387217816351611038419408862458817744394519270860576761494092006531204226033584127444589070143895166175437436668329261187900655405156900650251517330133691277945048148986936048852300859081938218299536012916904109770104679465447777558682844761741158553763956712306470952208024924857020367958829508871545858375036754029196314222346929519973568479114023250718057247670639419106042110366196596843535248667560253237016688158527945164395120202203676717219906311482758502081870551734123910863123241559269805744236887768653261714497292569358629512365063850421233459190154537628783024567287975838280735339385017217751444861230600109764523363874301956933912040997050780881454442690979636674545055825781307297617290973777867860936290498714105353419372883343485305903130145653616722465887749400776682935808648467175955974151729721523712022011769936770988660989862945168459185465259126425580092879314488216351985137564521067614437118421660055576572098630903746025458503018489103147462056648644597051626015482212386434089485520485853985656096706371345695920100777128278058279668245989164514304631730806242991532839575771483659845831600293913483401098730654084421756415488506576763637323270674854563084573287797304934047080754374469047218527012798272457579155457726484267047465276478571359000808679027397521380564320441826138519987902686308175306405031438783253338354635152626164345825423175627187165705895880457229674482502656372736429331868642738266891872788897361296821748514125957541781107008547394030853885717480397221973949478576700263132003970985246684186476004277133900234354929676579729351345293106268624673215173888445088900891034000041507040687871373532410274751569413475848139661411325084299168138660620533502402317454748600694292815750769731

```
679883496516946885520692142075287869829998451057763087443940161011144
9865790360315608261718876430408892194986527363696641095891021034109
1921152111715162119213636106072065886927677539955646631899211757271
1141428214630092334566935282312774777190985703140013134681462165696
84075228217378693978572221457284232503980240088440324523884462445644
54228389839738053070791981479771053758184730458505026058493430054931
74240718008728865820915751521295992388924371135403402786103092886806
62891359807863146467438587087138987377182513417125038345406237713399
245614161588559712455923868100669119042433950675217399865155800569961
0842627785219437167429983874012586028446291653667432138941025736017
93019105553112024269920462570810040405639066281616329134681855340279
02696993922723023427758701940228419852586243693272833271886448145579
22149141011536000062445258117160665996258772585932372029562279160005
82789305639705382637273046856479961741271496242635810082015138123980
10637962920440782246035301375626937703131613402998709859626790241918
44899803610621578361741855456364272781077822048780849047260914150171
13530738548236277178177764688268353806615431888765324260388991275829
98261328361264279731595611664454260158237827976439635342688219384578
94867243094724176661557515012848101542579830259964523371330169600241
68275473368262531014702762579325479344739430340966253180712095176378
73372059973111702494301577102163990145081814333817577997911944129519
57128837706041063464109596890435265859640075909154720740527652931664
60981865777187164560004521616713466202212148693436349079137155828749
92948061270853515483176734662611394863278795822045612420199631659400
86561516061771106726857141667633310109936028920594351395123735169927
51026636207349479845200952134029;

n :=p*q; //RSA modulus is computed
pn:=(p-1)*(q-1);  // Euler’s totient function Φ(n) for key generation
out:=GCD(e,pn); // To satisfy gcd(e, Φ(n))=1
while out ne 1 do
e:=Random(pn);
out:=GCD(e,pn);
end while;
d:=Modinv(e,pn); //Secret key computation
Cputime(initialization); // Measurement of initialization time

// Election
encryption:=Cputime();
total_S:=0;
product_C:=1;
// product_M:=1;
// product_M_NoMod:=1;
for k in [1..N] do
repeat
```

```
msg:=RandomPrime(J);
until msg ne 2;
s:= Random (1,Sbit);
total_S:=total_S+s;
r_msg:= msg*2^s; //Randomized message
c:=Modexp(r_msg,e, n); //Encryption of randomized message
// product_M:= (product_M*msg) mod n;
// product_M_NoMod:= (product_M_NoMod*msg);
product_C:=(product_C*c) mod n; //Encrypted & randomized vote product, EVPran
end for;
Cputime(encryption);

// Decryption of EVP
decryption:= Cputime();
//Encrypted_R:=Modexp(2^total_S,e, n);
//EVP:= (Modinv(Encrypted_R,n) * product_C) mod n; //Randomness cancellation
before decryption
//VP:=Modexp(EVP,d,n); //Decryption of EVP gives the Vote Product VP
VPran:=Modexp(product_C,d,n); //Decryption of EVPran gives VPran
VP:= (Modinv(2^total_S,n) * VPran) mod n; //Randomness cancellation after
decryption
VP;
product_M; // For a check with VP
finalNdivisions:= Cputime();
v2:=0;
v3:=0;
v5:=0;
v7:=0;
v11:=0;
v13:=0;
repeat
if (VP mod 2) eq 0 then
VP:= VP div 2;
v2:= v2+1;
end if;
until VP mod 2 ne 0;
repeat
if (VP mod 3) eq 0 then
VP:= VP div 3;
v3:= v3+1;
end if;
until VP mod 3 ne 0;
repeat
if (VP mod 5) eq 0 then
VP:= VP div 5;
```

```
v5:= v5+1;
end if;
until VP mod 5 ne 0;
repeat
if (VP mod 7) eq 0 then
VP:= VP div 7;
v7:= v7+1;
end if;
until VP mod 7 ne 0;
repeat
if (VP mod 11) eq 0 then
VP:= VP div 11;
v11:= v11+1;
end if;
until VP mod 11 ne 0;
repeat
if (VP mod 13) eq 0 then
VP:= VP div 13;
v13:= v13+1;
end if;
until VP mod 13 ne 0; // Vote counts are all found
v2;
v3;
v5;
v7;
v11;
v13;
e;
Cputime(finalNdivisions);
Cputime(decryption);
Cputime(total);
```

2. El Gamal Tallying with 1024-bit Modulus and *N*=256 Voters

```
// Initialization
initialization:= Cputime();
p_bit:=1024;
J:=4; //Bit size of candidate's prime number
N:=p_bit div J; // Number of voters
p := RandomPrime(p_bit);
// f := Factorization(p-1);
// f;
x:= Random(p); // Secret key
y:= Modexp(2,x, p); // Public key
```

```
Cputime(initialization);

// Election
encryption:=Cputime();
product_C1:=1;
product_C2:=1;
// product_M:=1;
for k in [1..N] do
repeat
msg:=RandomPrime(J);
until msg ne 2;
r:= Random(p); // Random number
c1:=Modexp(2, r, p); //Encryption, first entry alpha
c2:=msg*Modexp(y, r, p); //Encryption, second entry beta
// AlphaPowerx:= Modexp(c1,x, p);
// InvAlphaPowerx:=Modinv(AlphaPowerx,p);
// m:=(c2* InvAlphaPowerx) mod p;
// product_M:= (product_M*msg) mod p;
product_C1:=(product_C1*c1) mod p; //First entry of encrypted vote product, EVPran
product_C2:=(product_C2*c2) mod p; //2nd entry of encrypted vote product, EVPran
end for;
Cputime(encryption);

// Decryption of EVPran
decryption:= Cputime();
AlphaPowerx:= Modexp(product_C1, x, p); // EVP
AlphaPowerx;
InvAlphaPowerx:=Modinv(AlphaPowerx, p);
InvAlphaPowerx;
VP:=( product_C2* InvAlphaPowerx) mod p; //Decryption of EVP
VP;
// product_M; // For a check with VP

finalNdivisions:= Cputime();
v2:=0;
v3:=0;
v5:=0;
v7:=0;
v11:=0;
v13:=0;
repeat
if (VP mod 2) eq 0 then
VP:= VP div 2;
v2:= v2+1;
end if;
```

```
until VP mod 2 ne 0;
repeat
if (VP mod 3) eq 0 then
VP:= VP div 3;
v3:= v3+1;
end if;
until VP mod 3 ne 0;
repeat
if (VP mod 5) eq 0 then
VP:= VP div 5;
v5:= v5+1;
end if;
until VP mod 5 ne 0;
repeat
if (VP mod 7) eq 0 then
VP:= VP div 7;
v7:= v7+1;
end if;
until VP mod 7 ne 0;
repeat
if (VP mod 11) eq 0 then
VP:= VP div 11;
v11:= v11+1;
end if;
until VP mod 11 ne 0;
repeat
if (VP mod 13) eq 0 then
VP:= VP div 13;
v13:= v13+1;
end if;
until VP mod 13 ne 0; // Vote counts are all found
v2;
v3;
v5;
v7;
v11;
v13;
Cputime(finalNdivisions);
Cputime(decryption);
```

APPENDIX D

Proof of Non-Uniqueness of Vote Products Exceeding the Modulus

If we don't use the restriction $N < \log_2 n / J$ that assures $VP < n$, and let $VP > n$, is it possible to find the vote product uniquely using $D(E(VP)) \in Z_n^*$ and solving

$$D(E(VP)) + kn = VP \qquad \text{(D.1)}$$

for some value of k and some set of vote counts

$$v_1 + v_2 + \dots + v_C = N \,? \qquad \text{(D.2)}$$

In other words, is there a unique solution of (D.1) and (D.2) in terms of $v_1, v_2, \dots, v_C$ and k ? The answer is "No", which can be proved by the counter example below.

Counter example to uniqueness: Let $n=5\times 13=65$, $N = 6$ and $VP = 486 = 2\times 3^5$, with SCP= {2,3} assigned to 2 candidates. Since 486=31 (mod 65), $D(E(VP))=31$. Can we arrive at the valid factorization 2×3^5 using equations (D.1) and (D.2)?

Notice that $31+65=96 = 2^5\times 3 = VP_1$ satisfies (D.1) with $k=1$, and $N=6$ satisfies (D.2); similarly $31+7\times 65 = 486 = 2\times 3^5 = VP_2$ satisfies (D.1) with $k=7$, and also $N=6$.

So, nobody knows whether VP equals VP_1 or VP_2. This is why one should choose large enough bit size for the modulus such that $VP< n$ and the vote product must not exceed the modulus of the multiplicative homomorphic public key algorithm.

APPENDIX E

Speed of Modular Operations

We have first measured the CPU times of the 100,000 modular operations by using the Magma library on a 1,83 GHz CPU, and shown the results in Tables E.1 and E.2.

Table E.1 CPU times (in seconds) of the modular multiplication, inversion and exponentiation corresponding to 100,000 operations with 128 and 256-bit moduli.

Modulus Bit Size	Bit Size of Multiplicative Group Elements			Mult .	Mult.	Inv.	Exp.	Exp.
	g	r	e	$g \times r$	$r \times e$	r^{-1}	g^{r}	g^{e}
128	20	20	17	0.078	0.078	0.25	3.51	2.153
	40	40	17	0.078	0.078	0.421	6.942	2.324
	80	80	17	0.172	0.093	0.64	14.43	2.403
	160	160	17	0.312	0.172	1.014	31.652	2.558
	320	320	17	0.702	0.265	1.232	67.518	2.668
	Approximate ratio:			2	1	6	150	25
256	20	20	17	0.078	0.078	0.343	8.689	6.006
	40	40	17	0.078	0.078	0.453	18.611	6.474
	80	80	17	0.093	0.078	0.842	40.717	6.973
	160	160	17	0.265	0.078	1.435	90.871	7.691
	320	320	17	0.733	0.234	2.48	215.141	8.019
	Approximate ratio:			1	1	18	1165	100

We observe from Table E.1 and Table E.2 that exponentiation is the most time consuming operation. Inversion operation is the second and multiplication is the third but both are observed to be negligible with respect to exponentiation. In a group of given modulus, time for the exponentiation depends strongly on the bit size of the exponent (see the g^r column and compare with the last one g^e); and if the exponent size is doubled, exponentiation time is observed to be approximately doubled.

Table E.2 CPU times (in seconds) of the modular multiplication, inversion and exponentiation corresponding to 100,000 operations with 512 and 1024-bit moduli.

Modulus Bit Size	**Bit Size of Multiplicative Group Elements**			**Mult.**	**Mult.**	**Inv.**	**Exp.**	**Exp.**
	g	r	e	$g \times r$	$r \times e$	r^{-1}	g^r	g^e
512	20	20	17	0.063	0.078	0.406	25.397	18.626
	40	40	17	0.093	0.078	0.64	59.53	20.283
	80	80	17	0.094	0.109	1.061	129.263	22.074
	160	160	17	0.156	0.078	1.919	277.838	23.822
	320	320	17	0.608	0.078	3.806	664.471	26.613
	640	640	17	2.356	0.39	6.302	1427.315	28.034
	1280	1280	17	6.724	1.56	7.784	299.381×10	28.767
	Approximate ratio:			8	1	50	8500	340
1024	20	20	17	0.078	0.094	0.592	79.857	57.674
	40	40	17	0.078	0.078	0.874	197.154	63.43
	80	80	17	0.125	0.093	1.56	431.295	69.311
	160	160	17	0.156	0.078	2.402	926.163	75.786
	320	320	17	0.312	0.093	4.899	207.95×10	81.183
	640	640	17	1.857	0.109	10.733	461.14×10	88.717
	1280	1280	17	7.972	1.03	20.748	1006.05×10	94.896
	Approximate ratio:			17	1	100	42300	800
				1/2488		1/423	1	

To find the dependence of the exponentiation speed on the bit size of the modulus, we compare the time for 80-bit exponentiations versus modulus bit size in Table E.3 and observe that as the modulus bit size is doubled, exponentiation time is tripled.

Table E.3 CPU times (in seconds) of the 100,000 modular exponentiations with 80-bit numbers for 128, 256, 512 and 1024-bit moduli.

Modulus Bit Size	**Bit Size of Multiplicative Group Elements**		**Exp. g^r**	**Approximate ratio with respect to the half-bit-size modulus**		
	g	***r***				
128	80	80	14.43	1		
256	80	80	40.717	2.8	1	
512	80	80	129.263		3.2	1
1024	80	80	431.295			3.3

To make this comparison more fairly, one should take into account that as the modulus size is doubled, the size of a randomly picked number is also doubled; hence we expect the corresponding exponentiation time to be multiplied by 6. Table E.4 confirms this expectation.

Table E.4 CPU times (in seconds) of the 100,000 modular exponentiations with comparable numbers for 128, 256, 512 and 1024-bit moduli.

Modulus Bit Size	**Bit Size of Multiplicative Group Elements**		**Exp. g^r**	**Approximate ratio with respect to the half-bit-size modulus**		
	g	***r***				
128	80	80	14.43	1		
256	160	160	90.871	6.3	1	
512	320	320	664.471		7.3	1
1024	640	640	4611.400			6.9