UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

AN ELASTIC ELECTRONIC VOTING SYSTEM

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MARIO SERGEI CORNEJO RAMIREZ

PROFESOR GUÍA:
TOMÁS BARROS ARANCIBIA
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
JOSÉ PIQUER GARDNER
PATRICIO POBLETE OLIVARES
JEROEN VAN DE GRAAF

Este trabajo ha sido parcialmente financiado por INRIA Chile y NIC Chile Research Labs

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: MARIO SERGEI CORNEJO RAMIREZ
FECHA: 2015
PROF. GUÍA: SR. TOMÁS BARROS ARANCIBIA
PROF. GUÍA: SR. ALEJANDRO HEVIA ANGULO

AN ELASTIC ELECTRONIC VOTING SYSTEM

En los últimos años, organizaciones y movimientos sociales han aparecido demandando más participación en políticas públicas. En éstas organizaciones, los miembros demandan ser parte del proceso de toma de decisiones el cual generalmente se realiza mediante iniciativas de voto directo entre los miembros. Además utilizan Internet intensamente como la plataforma principal de comunicación y tienden a confundir sistemas de encuestas con herramientas de votación electrónica. Por otra parte, no es claro que *software* se debe utilizar, y la debilidad de la mayoría de éstos que tienen con respecto a la robustez (capacidad de computar bien el resultado) más que con la privacidad del voto, además de la facilidad de uso.

Como una forma de mejorar la participación, nosotros proponemos un sistema de votación electrónica para ese segmento, que incluye organizaciones sociales, federaciones de estudiantes, colegios, sindicatos, sociedades profesionales, etc. El problema de votación electrónica ha sido ampliamente estudiado por criptógrafos, y hoy en día, existen varios protocolos para resolver problemas específicos a votación electrónica. Nosotros proponemos una solución que toma en consideración esas soluciones existentes combinadas con protocolos de sistemas distribuidos para introducir un sistema de votación electrónica remota elástica. El sistema utiliza la tecnología *elastic computing* de Amazon que permite escalar en términos de capacidad de computación y alta disponibilidad junto al anonimato de los votantes y la garantía que el voto fue correctamente contado. Concretamente, el sistema está pensado sobre cinco principios: i) Computación elástica, ii) Internet iii) Facilidad de uso, iv) Anonimato y computación verificable, v) Cliente liviano.

El objetivo de esta tesis no es solamente resolver el problema abierto descrito anteriormente, sino también establecer una base sólida para plataformas de votación electrónicas a través de Internet. De este modo, nosotros creamos un nuevo sistema de votación electrónica en donde el votante no realiza ninguna computación grande, sino que la trasladamos al servidor, que idealmente está en una plataforma de Cloud Computing como Amazon Web Services. Esta técnica previene ataques de denegación de servicio, robo de identidad y accesos no autorizados, al mismo tiempo preserva la privacidad y la verificabilidad.

La plataforma se probó en un caso real, concretamente en una experiencia de votación electrónica en donde los chilenos demandando su derecho a voto en el extranjero, pudieron votar en una elección simbólica. Se presenta la experiencia, los problemas y las soluciones que encontramos utilizando un sistema de identificación simple. Esta proyecto nos permitió estudiar de forma técnica, política y práctica aplicaciones de votación electrónica en América Latina.

Abstract

Over recent years, social organizations and movements have increased, demanding more participation in public policy issues. Within these organizations, members demand to be part of the decision making process, usually exercised via direct voting of the initiatives. These new organizations use internet intensively as the main communication platform and tend to confuse survey or poll systems with an e-voting tool. For this kind of organizations it is not clear which voting system should be used: privacy is not the weakest point of most existing and publicly available e-voting software, but rather soundness – being able to correctly compute the result – as well as flexibility and ease of use.

To improve participation we propose an electronic voting system for that kind of organizations, which includes social rights organizations, student federations, schools, professional associations, etc. The e-voting problem has been studied by cryptographers for a long time so that today, several protocols exist to solve specific issues. We propose a solution that uses some of these known protocols together with some distributed system techniques to introduce an elastic e-voting system. The system combines elastic computing and high availability with anonymity of the voters and the guarantee that the vote is correctly counted. Concretely, the system is designed under five principles: i) Elastic Computing ii) Open Network iii) Ease of use iv) Ballot privacy and tally correctness, and v) Lightweight client.

The goal of this thesis is not only to solve the open problem described above, but also to establish solid grounds for e-voting platforms using Internet. Thus, we create a new e-voting system where the client performs nothing but a small computation, and the heavy computation is on the server side, ideally deployed in a Cloud Computing platform like Amazon Web Services. This technique prevents attacks such as: denial-of-service, spoofing, and unauthorized intrusion while preserving privacy and verifiability.

We deployed our implementation in a real-world electronic voting experience where Chileans who demanding their right to vote from abroad were able to vote for presidential elections in Chile in a non-binding (symbolic) election. We present our experience, explain the problems, provide some solutions, and document the results of a real e-vote with simple identity validation. This project allowed us to study the technical, political and practical applications of electronic voting in Latin America.

Thank you for all. I am never going to forget everything what you did.

Acknowledgements

Over the years I met a lot of marvelous people that changed the way of how I see and live the life. I would like to thank all of you and let you know that you were always in my thoughts. Thank you for helping me get through the adversity and for being with me during the good and the bad times.

To my advisors and friends Tomás Barros and Alejandro Hevia, thank you for all the time you invested in me and all the discussions that we had. Thank you for showing me the world of professional research with all its challenges, benefits, and drawbacks. It has been a privilege working with you. Tomás, thank you for the trust and confidence in me, for having accepted me in the NIC Labs and having given me the necessary funding. Without this, it would not have been possible for me to pursue my master's.

Thanks to all my colleagues that I have been working with in NIC Labs, INRIA and E-Voting. To Claude Puech for his continuous positive enforcement to keep up my studies in Chile and abroad, José Miguel Piquer for showing me that the academic world can be mixed with real life engineering, my friends Pablo, Pamela, Adrían, Raúl and Sebastián for pushing me to continue even in bad times.

David Pointcheval and Michel Abdalla, thank you for having accepted me for my internship in the prestigious École Normale Supérieure de Paris, an experience that really touched me and changed my life. A special thanks to Antonia Schmidt-Lademann for her friendship, time invested, valuable comments and corrections to this work.

I am very grateful to all people of the Computer Science Department (DCC) at the University of Chile as I learned so many things apart from my studies. I thank my fellow students and friends Milton Inostroza, Matías Toro for all the talks and time that we spent together. I am also grateful to the professors and administrative staff, in particular Angélica Aguirre and Sandra Gaez, thanks to all for your support.

Last but not least, I thank Rubén González, Francisco Chávez, Víctor Ramiro, Rossana Escobar, my parents Consuelo Ramírez and Pedro Cornejo and all my family for the titanic task of supporting me all this time and encouraging me to keep working hard. I dedicate this work to all of you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Elections are among the most important processes in modern democracies. Even current monarchies still use elections to choose their prime minister or members of the parliament. Each nation has its own system to select their representatives by popular vote. Paper-based voting systems are the most commonly used, in this type of systems voters mark their preference on paper, seal it and put it in a box to be counted at the end of the election.

The development of modern cryptography has driven improvements in different areas, changing aspects of our lives in many ways. We can communicate instantly and securely send photos and videos from one part of the world to another. We can pay using our credit cards remotely or control nuclear power plants from a distance. Even with all the new development, the electronic voting problem is still open and we are far from reaching a consensus.

The electronic voting (e-voting) could be seen as the modernization of the current paper-based voting, but opinion on this topic is divided. Some believe that it is not possible to achieve privacy and confidence with the current technology, while others feel that e-voting should already be in use. Currently, there are three ways to cast a vote electronically: supervised precinct based on local tally, supervised precinct based on remote tally and a remotely based (unsupervised). The first option tries to capture the idea of traditional voting but uses a machine instead of paper. The remotely based version casts a vote from anywhere, in particular from houses using a personal computer or from anywhere using a mobile smartphone.

The introduction of electronic voting produces benefits in the usability; adding more information to help voters, reducing the paper-vote logistics, and introducing a new tally method which can reduce errors in the sum. These errors in the tally process might be intentional or not, like counting a vote for a different preference than is marked on the ballot, counting blank votes (or marking them) for a particular preference and marking more than one preference in order to spoil it. This misbehavior is likely to occur in isolated voting precincts or when the judges of the election are colluded for one candidate. But the introduction of technology also faces some cons like computational security concerns.

1.1 Motivation

Nowadays, there is a worldwide demand for more participation in national politics from society associations that reach across formal institutions of decisions (political parties, assemblies, commissions etc.). Some examples of this are the so-called "Arab Spring" in the Middle East, North Africa and Central Asia [SW12], "Los Indignados"[Alc11][Rai11], "Yo soy 132" [Mat12] or "Occupy Wall Street"[Gab11][BBC11].

The same phenomenon has been occurring in Chile since the "Revolución Pingüina" [Sca06] [Del11] [Bar11] that triggers mass population demands. Since then, several organizations have or have had a significant increase in participation. Some of them are "Voto Ciudadano"[Vot14], "Ciudadano Inteligente"[Ciu14], "Revolución Democrática"[Rev14] or "Fundación Iguales"[Fun14]. In all those organizations, members demand to be part of the decision making process, usually exercised via direct voting of the initiatives.

This new democracy is highly participative, with a variable voting universe, made up of a few people to hundreds of thousands. To stimulate participation, the organizations themselves use new communication platforms and technologies such as Twitter, Facebook, WhatsApp, etc.[Tay11] [Man12] [Jur12] [Pas12]. Their members use new social media and Internet communication platforms to enable the assembling of people with diverse interests but a common goal.

Most of these organizations regularly call for a vote on the decisions and actions they plan to take. In October 2014, the City of Santiago conducted a communal consultation on local topics. In that consultation more than $19,000$ votes were received only via Internet, whereby several problems occurred. First, the consultation had no kind of privacy properties, the vote was not encrypted and to vote, only the ID number was required to cast a ballot. A redundancy platform was not created, and at peak hours the consultation was down because of a denial-of-service caused by the good turnout and interest in vote.

In this attempt at electronic voting they intended to achieve a binding vote through a in-house development, mainly because most of the available electronic voting platforms are not easy to use and deploy for non-experts in security or cryptography.

1.1.1 Design criteria

In this work, we propose a voting system considering the previous experience of multiple organizations to satisfy their multiple requirements. From our experience, the most important requirement is to assure that voters cast a ballot during the period of time while the voting is open. Any downtime of the voting system for any reason will generate people's a mistrust, and the entire election could be in jeopardy. Based on this, we define five fundamental principles, which we detail next.

We implement High Reliability through an **Elastic Computing** architecture allowing linear scale on demand without later configuration. A computer architecture is said elastic

if the computational power can be increased or decreased on demand. Maintaining a zero-downtime is critical to a voting process, to ensure that every voter who wants to participate in an election casting their vote can do so without problems. Outsourcing the infrastructure is a convenient alternative for organizations that do not have the required hardware and want to call an election for two or three days. Distributing the servers between one or more clouds prevents a single point of failure as well as Denial-of-Service (DOS) attacks. Since we do not control the hardware, the next requirement becomes a key issue that must be taken into consideration.

In the requirement **Secrecy and Tally correctness** we focus on an integral model against an active adversary. The platform maintains ballot privacy and verifiability (open-audit) using cryptographic techniques such as vote encryption, signatures and distributed authorities. To preserve the secrecy of the vote we rely on homomorphic encryption to securely compute the tally as the sum of the cast votes, meaning that only the result gets decrypted while all the votes remain encrypted. The Decryption key is securely generated and distributed among n authorities. In order to decrypt the result, t out of n shares are needed (preventing collusion and no-cooperation of a small part of the authorities).

As we introduce a remote voting platform, an **Open Network** requirement is compulsory. We explore the possibility of using the Internet with no specific infrastructure for electronic voting. Voters and authorities only need to use a standard SSL/TLS connection. The voter authentication mechanism simple: user/password or pseudo-secret information already existent on the national identity document. Open Network reduces the entrance barriers of implementing a VPN for every node that wants to vote.

The requirement **Easy to Use** means providing a voting platform for people who are not IT specialists. This platform is intended to be easy to use and for that reason, it has a friendly-user web portal for all the participants involved in the election. Anyone can use the platform without technical knowledge. It is possible to deploy the entire architecture with just a few interactions. Though this might obvious, a voter just has to mark the option and cast the vote, but underneath there is a lot more things to take care. For the electoral committee that supervises the complete process, this criterion means that they can create an election in three simple steps. Each authority has a digital key that allows him to proceed with the count and opening of the votes. The authorities receive their respective key in an off-line way by using a USB stick or similar, and shortly after the platform is ready to cast ballots. Once the election is finished, the authorities upload their share of the key and wait while the system collects all shares and performs the decryption of the tally. Setting up an election requires only three decisions to make. First, selecting the basic information such as the election title, the questions, and opening and closing dates. Second, selecting the electoral roll or participants and, selecting the judges (authorities). For voters this is also easy. A voter first encrypts the vote on one of our many encryption servers and then the votes himself casts it. The voter only authenticates for ballot casting, in this way, the ballot does not contain any personal information from the voter.

The last goal is to use a **Lightweight client**. We move all the heavy computation to the server-side, moving cryptography from the client to the elastic cloud. Encryption, verification and tally are performed in the cloud.

1.2 Organization of the thesis

In Chapter 2, we present the background of our work, related work, current security models and implementations. In Chapter 3, we explain all the cryptographic tools used, the basis and the mathematics involved in our work. In Chapter 4, we introduce our remote electronic voting system, defining the entities and the different steps involved in the complete process. In Chapter 5, we explain the implementation and technical decisions considered. In Chapter 6, we explain all our technical, practical and cryptographic assumptions. Finally in Chapter 7 we present in detail our experience using this work in a real-world election.

Chapter 2

Background

2.1 Related Work

Theoretical results from the past 30 years generated by a significant number of computer scientists and mathematicians led to the development of multiple protocols like encryption schemes, signature schemes and zero knowledge proofs, among many others. These have allowed secure communication protocols like SSL, VPN and so on. With all of those protocols created, electronic voting emerges in order to modernize the current paper based voting process.

Electronic voting research began in the late 80's when researchers like Josh Cohen (Benaloh) [Ben87] and David Chaum [Cha88] proposed the first provably secure schemes for electronic voting. These voting systems preserve the ballot secrecy, and privacy. That means that no other participant than the voter himself should be able to determine the value of the vote cast (ballot) by this voter. The intuition is that the ballot has to be encrypted or ravelled.

In [Ben87] Josh Benaloh developed one of the earliest practical voting schemes, fully implementable, preserving ballot secrecy and incorporating verifiability for all the participants and mere observers who are all convinced that the tally is correct. As far as we know, this scheme was the first to use the secret sharing [Sha79] (threshold) concept, allowing direct computation on the shares altogether to prevent the corruption of a small set of authorities.

David Chaum in [Cha88] proposes an election protocol that ensures an unconditionally-secret ballot and correctly ballots count based on the low probability to learn the dth roots modulo N. Moreover, Chaum claims that it is untraceable to disrupt the election, based on the RSA [RSA78] assumption.

Later, in the 1994, Benaloh and Tunistra in [BT94] present a new idea eliminating any kind of receipts which can be used to prove to others how a specific person voted. This concept is called *Receipt-Freeness*. Benaloh *et al.* propose the use of a *beacon* which produces an unlimited supply of unpredictable bits and a homomorphic encryption scheme. The authority

sends some values to the voter over a private channel and the voter emits the vote over a public channel. Later, Hirt and Sako [HS00a] show that this scheme in fact is not receipt-free.

One year later, Niemi and Renvall [NR94] publish the same concept in an independent work, satisfying both properties: Each voter can verify the tally and the voter cannot prove the vote. In 1996, Okamoto [Oka96] proposes a practical large scaled secret voting scheme which satisfies the property that there is no receipt of a vote, to buy a vote or coerce a voter. One year after that, the same Okamoto in [Oka97] himself proves that his previous work was not receipt-free indeed. In these subsequent works Okamoto uses blind signatures in order to achieve receipt-freeness.

Sako and Kilian in [SK95] make an advance in order to implement receipt-freeness in a more practical manner. They attack the problem of the voting booth, using one physical assumption: The existence of an untappable private channel that cannot be achieved as a cryptographic implementation must be physical. This assumption is strong for a practical solution. In this scheme, Sako et al. propose a distributed authority (making it more difficult to disrupt the entire election) and an efficient tally based on homomorphic encryption. Further, it is the first work that uses a mixed network introduced in [Cha81] to achieve universal verifiability. The work of Sako and Kilian was used by Hirt and Sako [HS00b] in order to accomplish one of the most efficient and correct receipt-free voting schemes.

Mixnet was used in Park's *et al.* previous work [PIK94] and Fujioka *et al.* [FOO92]. These schemes introduce optimizations. However, they just achieve individual verifiability.

The authors who rely in these assumptions generally claim that their schemes are practical, but in reality they are not. The idea behind it is to provide perfect secrecy in an information-theoretic sense, while in a large-scale or national scope voting it is difficult to cover all the polling booths.

Jules, Catalano and Jakobsson in [JCJ05] introduce a new security model for e-voting which is more powerful than what can be found in previous work. They include new threats like vote buying which only requires an anonymous channel and avoids using an untappable channel. In order to achieve this new strong security model, Jules *et al.* introduce a credential which is a secret value that is unique to the voter to ensure that the ballots are cast by legitimate voters. They also introduce a fake credential to impede vote-buying and to defraud any coercer.

The work of Benaloh [Ben06] introduces a detailed e-voting protocol which offers a verifiable election system that is far simpler than other such systems previously proposed. Benaloh divides the different processes involved in a election into parts and mixes them with cryptographic protocols without proposing any implementation. Gardner, Garera and Rubin in [GGR09] extend the Benaloh scheme and present a new definition of coercion which is more likely to resist cryptographic security games for end-to-end voting. Using the Benaloh scheme, Gardner *et al.* implement the whole protocol with formal and provable secure definitions.

A new game-based security model using formal methods (symbolic model), was presented by Küsters, Truderung and Vogt in [KTV10]. In this model Küsters *et al.* define objectives (or goals) and the probability of achieving these goals with one or more strategies. In par-

ticular, the objective is to avoid coercion. Using this technique it is also possible to measure the “level of coercion" by the different schemes.

Electronic voting has developed gradually, from the first scheme presented by Benaloh and Chaum to the model of Küsters. All the security models presented here try to capture the ideas of paper-based voting model, using cryptographic assumptions. The experience of implementing electronic voting in the real world seems distant from the academic world since the voters have different concerns. For example, voters are more interested in ballot secrecy than coercion. Having mechanisms to avoid coercion like having two pairs of keys with the aim of cheating on the coercer are usually complicated to implement and might induce errors on behalf of the voter who does not necessarily understand how it works.

In practice voters do not have (and they do not want to have) a key pair to vote or remember cryptographically strong passwords. In fast, having this pair of keys often creates more problems than it solves [And01]. In real life, voters are not computer specialists and it is difficult for them to manage their private keys and certificates [BWB05]. In this work, we intend to relax some of these security properties to create a usable electronic voting system.

In the next section we present the definition of voting schemes similarly to [BCP+11]. We define a general syntax for the major voting schemes.

2.2 Generic voting scheme

An election system may consist of several sets of entities as presented in [JCJ05] :

1. **Registrars:** Denoted by $\mathcal{R} = R_1, R_2, \ldots, R_{nR}$, this is a set of nR entities responsible for jointly issuing keying material.
2. **Authorities:** Denoted by $\mathcal{T} = T_1, T_2, \ldots, T_{nT}$, this is a set of nT authorities responsible for processing ballots, jointly counting votes and publishing a final tally.
3. **Voters:** The set of n_V voters, denoted by $\mathcal{V} = V_1, V_2, \ldots, V_{nV}$ are the entities participating in a given election administrated by $\mathcal{R}$.

A Bulletin Board $\mathcal{BB}$ is defined as a piece of universally accessible memory to which all players have append-write access. In this memory, any player can write data to $\mathcal{BB}$ but cannot overwrite or erase the existing data. This is used to publish all the ballots cast by voters. Also, a candidate slate $\mathcal{C}$ is published which contains all possible candidates or voting options.

A voting scheme can be defined by the five generic algorithms *Setup, Registering, Vote, Tally, Verifying*, where each one encapsulates one function of the scheme.

1. **Setup:** The setup algorithm takes as input a security parameter 1^λ and $\mathcal{R}$, returns a secret information SK_R (may be shared by all registrars) and initializes a new bulletin board for the election. We write *(SK_R, $\mathcal{BB}$)* $\leftarrow$ *Setup(1^λ, $\mathcal{R}$).*

2. **Registering:** The registering algorithm assigns the right to vote to the voters $\mathcal{V}$ and selects the authorities $\mathcal{T}$ for one particular voting. For each entity, the function *Register* creates the credentials to *log in* or validate oneself depending on the voting protocol, and also returns the public key of the particular voting created. We write *(*SK_{Vi}*,* SK_{Ti}*,* PK*)* $\leftarrow$ *Register(*$\mathcal{V}$*,* $\mathcal{T}$*,* SK_R*).*
3. **Vote:** The voting algorithm takes a valid option from the candidate slate and the public information, and creates a ballot which is sent to the bulletin board. We write *(b)* $\leftarrow$ *Vote(v,* PK*).*
4. **Tally:** The tallying algorithm cleans all the spurious votes, computes the vote tally with all the valid votes $\mathcal{X}$ and creates a non-interactive proof $\mathcal{P}$ that the tally was correctly computed. We write *(*$\mathcal{X}, \mathcal{P}$*)* $\leftarrow$ *Tally(*$\mathcal{BB}$*,* SK_T*,* $\mathcal{C}$*).*
5. **Verifying:** The verifying algorithm takes the final tally, the bulletin board and the non- interactive proof in order to check if the tally was correctly computed. We write $\{1, 0\} \leftarrow$ *Verify(*PK_T*,* $\mathcal{BB}$*,* $\mathcal{C}$*,* $\mathcal{X}$*,* $\mathcal{P}$*).*

In the next section we define some properties of the e-voting protocols presented in the literature.

2.3 Basic Properties

The verifiability allow voters and election observers to verify that the votes has been recorded, tallied and declared correctly.

Individual Verifiability: The voter should be able to verify that his intentions were accurately recorded in the cast ballot and check that this ballot was published on the bulletin board.

Universal Verifiability: Voters should be able to verify that all cast ballots were properly included in the final tallies and came from legitimate voters. Anyone can check that all the votes in the election outcome correspond to the ballots published on the bulletin board.

Eligibility: Anyone can check that each ballot published on the bulletin board was cast by a registered voter and counted at most one time.

Mandatory Privacy: No one should be able to learn how another voter voted with certainty, even if the voter would like to let that person know.

Fairness; This prohibits the voting system from influencing a voter's behavior, that is, the observation of the voting does not leak information.

Ballot independence: Observing another voter's interaction with the election system does not allow a voter to cast a meaningfully related vote.

No early results: A voter cannot change his vote once partial results are available.

Pulling out: Once partial results are available a voter cannot abort.

In the next section we define security models for e-voting protocols:

2.4 Security models

In this section we present the different security models defined by the authors in the literature starting by the weak model to the currently strongest. The different security models ensure preventing intimidation of voters and that voters can express their free will without fear of retribution.

Ballot privacy Introduced by Benaloh and Tunistra [BT94] the intuition is that a voter's vote is not revealed to anybody. So, ballot secrecy requires that the adversary is unable to distinguish between real ballots and fake ballots. In [CS13] they define ballot secrecy as the assertion that an adversary (controlling an arbitrary number of dishonest voters) cannot distinguish between a situation in which voter A votes for candidate x and voter B votes for candidate x', from another situation in which A votes x' and B votes x. Cortier *et al.* expresses this by the following equivalence:

$$A(x) \mid B(x') \approx_l A(x') \mid B(x)$$

Another model to express ballot privacy is presented by Bernhard *et al.* in [BCP+11]

Receipt freeness The concept was introduced in 1996 and refined by Okamoto in [Oka97]. The intuition is that the voting system generates no receipt of who a voter voted for, because the receipt of a vote proves that a voter has voted for a candidate and could therefore be used by another party to coerce the voter.

By this simple concept it is possible to achieve a stronger model that implies the previous one. Okamoto defines the receipt-freeness based on the following simulation, which can still be used based on our voting scheme definition in section 2.2:

Given published information X (public parameters and bulletin board), adversary C interactively communicates with voter V_i to cast C's favorite vote v_i^* to $\mathcal{T}$, and finally C decides whether to accept $View_C(X : V_\mathrm{i})$ or not, and $\mathcal{T}$ decides whether $\mathcal{T}$ accept v_i^* or not. Here, C gets the message x_b from the bulletin board $\mathcal{BB}$ immediately after x_b is put on the board (C can see it from $\mathcal{BB}$ because it is public). $View_C(X : V_\mathrm{i})$ means C's view through communicating with V_i and getting information from the bulletin board, that is $View_C(X : V_\mathrm{i})$ includes published information X, C's coin flips, v_i^*, and the messages that C receives from V_i.

A voting system is receipt-free, if there exists a voter V_i such that, for any adversary C, V_i can cast $v_\mathrm{i}(v_\mathrm{i} \neq v_\mathrm{i}^*)$ which is accepted by $\mathcal{T}$, under the condition that $View_C(X : V_\mathrm{i})$ is accepted by C.

Coercion resistant Introduced in [JCJ05], Juels *et al.* define an adversary that cannot interact with voters during the election process. In particular, an election is private if such an adversary cannot guess the vote of any voters better than an adversarial algorithm whose only input is the election tally. The coercion resistance is a strong form of privacy in which it is assumed that the adversary may interact with voters. To our knowledge, the coercion resistance captures the fullest possible range of adversarial behavior in a real-world, Internet-based voting scheme. It offers receipt-freeness and defines a scheme against randomization, forced-abstention and simulation attacks all in the face of corruption of a minority of the tallying authorities.

The current security models of coercion-resistance are tailored to each protocol. Like [CCM08] we use an informal definition of coercion-resistance. In [JCJ05] and [DKRD06] two ways are presented to formally define coercion-resistance.

In the next section we present some of the current implementations.

2.5 Implementations

2.5.1 Remote voting solutions

Helios

The work by Ben Adida in [Adi08] has been used by several institutions [1] like universities and associations like the IACR in order to vote for student governments or committee members.

Helios (version 3) is an open source, web-based, single server platform implemented in Python with several desirable features for an internet voting system. Helios permits logging-in with external providers like Google, Twitter or Facebook. There are flexible registration lists for closed or open elections. It is based on a threshold key authority distribution to

[1] As it is said in the webpage of the project. http://heliosvoting.org/

enable a distributed decryption by multiple trustees, and provides yet other features.

The Helios voting scheme is based on Benaloh's scheme and Sako-Kilian's mixnet. It is similar to our generic scheme presented in section 2.2 implemented with the Elgamal encryption scheme, supporting semantic security, re-encryption, and a non-interactive proof for the correct mix and a proof of decryption.

Civitas

Michael Clarkson *et al.* in [CCM08] presents the first electronic voting system that is coercion-resistant, universally and voter verifiable, and suitable for remote voting. Civitas is a provably secure voter registration protocol, with distributed trust over a set of authorities and a scalable design for vote storage that ensures integrity. The implementation of [CCM08] has coercion-resistant with a ranked voting method.

Civitas uses a publicity available specification of cryptographic protocols to implement a coercion-resistant, verifiable and remote voting scheme. This implementation is written in Jit and it is based on the scheme by Juels et al., with some differences. All the voters have two keys, a registration key and a designation key: The first one for authenticating the voter and the other to acquire the private credential which must be sent with the vote (both encrypted).

Cryptographic components used by Civitas are standard protocols, a public key infrastructure to encrypt and sign messages, and a variety of zero-knowledge proofs to enforce the honest execution of the protocols.

Following the generic scheme presented in this work, the Registrars use RSA keys and the authorities generate a distributed Elgamal key. The Registrars post each voter's registration public key and each designation public key.

The voters encrypt their votes using Elgamal which is homomorphic with respect to multiplication. To avoid the malleability (inherent in homomorphic encryption) Schnorr signatures are required. To ensure the honesty of the Authority, zero knowledge proofs are generated during key generation and decryption. Once the voter is authenticated, a shared AES session key is established between them. Besides the ballot, two zero-knowledge proofs are generated, *1-out-of-L* and *proof of knowledge of two values*.

2.5.2 Estonia Electronic System Platform

Estonia was a pioneer in the world to use Internet voting nationally, and today more than 30% of its ballots are cast using this platform called I-Voting. Estonia has developed a national full Public Key Infrastructure which can be used for voting. Each citizen has a key pair in their own national ID card and to cast a vote, they download a desktop application that the voter connects to the central system to store the votes into the database. Springall, Finkenauer, Durumeric, Kitcat, Hursti, MacAlpine, and Halderman[SFD+14] show several

flaws in the process, focused in how the process was carried out, and the desktop application than the security of the scheme.

2.5.3 Precinct-based solutions

Puchscan

Kevin Fischer *et al.* provides a clear definition of the Puchscan concept in [FCS] introduced by David Chaum. It is a hybrid paper/electronic system. It employs a two-layer ballot and receipt, and a sophisticated cryptographic tabulation system. This implementation uses ballots with two layers: The voter marks the option with an ink dauber and separates the layers.

The system saves information about the two layer ballot and compares it with the cast ballot in order to reconstruct the option.

Wombat voting

In [BnFL$^+$12], Ben-Nun *et al.* presents a dual voting system (paper and electronic) designed to mix current electronic voting techniques with traditional paper based ones. This voting platform is an adaption of Benaloh's system [Ben06] to Israel's paper-based system. The voter identifies himself at the voting booth and marks a preference in a voting machine with touch screen. This voting machine prints a divided receipt. On once side it shows the vote in plaintext and on the other it contains the encrypted and signed vote. The plaintext is cast in a traditional way, while the other part is scanned and uploaded into the bulletin board. Using this simple technique it is possible to verify all ballots reading the plaintext and compute the tally using the homomorphic characteristics.

2.5.4 Others

There are other implementations such as ThreeBallot, Scantegrity II, Prêt à Voter, among others, which are not purely designed for remote voting. Those implementations mix paper based with electronic voting protocols. Most of them use the paper part to generate security properties which are used in the security of the model.

Chapter 3

Cryptographic tools

Modern Cryptography is concerned with the construction of efficient schemes for which it is infeasible to violate the security feature. In electronic voting, several cryptographic and mathematical concepts are used to achieve security properties such as ballot privacy, anti collusion, high availability and verifiability. In this chapter we define the cryptographic tools used in this work, their roles and the technical terminology for a better understanding.

3.1 Basic Cryptographic Primitives

The basic setup for an encryption scheme is a *sender* who intends to transfer information to a *receiver* over an *insecure channel* that may be tapped by an *adversary*. The receiver must be capable of correctly obtaining the secret information meanwhile the adversary can not.

We define the information which the sender will send as *plaintext* and the ravelled information as *ciphertext*. The plaintext is private and only meant for sender and receiver, while the ciphertext is public. Sender and receiver have an algorithm to transform the plaintext into ciphertext, they both must be different and it should be difficult to convert the ciphertext into the plaintext without some secret information which is called *the decryption key*. An encryption scheme is defined as an algorithm that can perform the transformations of the plaintext into the ciphertext and the contrary using adequate keys.

The **encryption algorithm** takes the plaintext and the encryption key as input and returns the ciphertext. The **decryption algorithm** takes the ciphertext and the decryption key as input and return the plaintext. In order to create adequate keys for both algorithms a third algorithm **key generation algorithm** takes the size of the key and returns the secret/public key pair (encryption key, decryption key).

These three algorithms are called encryption schemes and are public for all participants including the adversary. The idea behind it is that the adversary does not know the decryption key. Those encryption schemes enable secure communication without the use of a *physically* secure channel.

3.1.1 Private-Key Cryptography

There are two types of this encryption scheme, depending on the encryption-key and decryption-key. If both are equal, the scheme is called symmetric or Private-Key. In this scheme, both parties have the same private-key (securely exchanged before the data transmission). The sender encrypts the plaintext using this key and sends it through a public channel. Then the receiver decrypts the ciphertext using the same key in order to recover the plaintext.

AES

The Advanced Encryption Standard (AES) is the encryption standard specification approved in 2001, proposed to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. It is based on several substitutions, permutations and linear transformations, each executed on data blocks of 16 bytes. Those operations are called rounds and repeated several times. During each round, a unique roundkey is calculated and used in the operations. Even if only a single bit is different in the input, the resulting ciphertext will be completely different. To encrypt messages whose length is longer than 128 bits, a block cipher must be used repeatedly following a procedure called *mode of operation* [FIP01].

3.1.2 Public-Key Cryptography

In the Public-Key Cryptography (also called asymmetric) the Key Generation Algorithm produces two different keys, a *public key* (PK) and a *secret key* (SK). An important property is that given the public key it is infeasible to forge the secret key. In this scheme, the receiver sends his public key to the sender over an insecure channel, who then encrypts the plaintext using that key and transmits the resulting ciphertext. Finally, the receiver can decrypt the ciphertext using his secret-key. In Public Key Cryptosystems both users exchange their public-keys publicly to encrypt a plaintext that can be decrypted only with the secret-key associated with each public-key. By publicly revealing PK one does not reveal an easy way to compute SK.

RSA

RSA is the first practicable public key cryptosystem proposed. It was designed by Rivest *et al.* in [RSA78]. based on Diffie and Hellman's previous work [DH76].

- **Key Generation** On input k as security parameter, choose two distinct primes p and q at random with $2^{(n-1/2)} \leq p, q \leq 2^{n/2}$. Calculate $n = pq$ such that N is a k-bit number and $\phi(n) = (p-1)(q-1)$ is the Euler's ϕ function. Choose e such that $1 \leq \mathrm{e} \leq \phi(n)$ and $gc\mathrm{d}(\mathrm{e}, \phi(n)) = 1$ (coprimes). Determine d as $\mathrm{d}^{-1} = \mathrm{e}(\mathrm{mod}\phi(n))$. The public key consists of the modulus n and the public exponent e. The private key consists of the modulus n and the private exponent d.

- **Encryption** On input $x \in Z_n$ and $PK = (n, \mathrm{e})$ perform $y = x^{\mathrm{e}} \bmod n$ to get the cipher text.
- **Decryption** On input $y \in Z_n$ and $SK = (n, \mathrm{d})$ calculate $x = y^{\mathrm{d}} \bmod n$ to recover the plaintext.

Paillier Cryptosystem

In particular, the voting scheme presented in this work is based on Paillier's Cryptosystem [Pai99] taking into consideration the optimizations made by Damgård *et al.* [DJN10]. Concretely, we use the encryption scheme CS which works as follows:

- **Key Generation** On input k as security parameter, choose an admissible RSA modulus $n = pq$ of length k bits. The public key is n and the secret key λ is the least common multiple of $(p-1)(q-1)$.
- **Encryption** Given $\mathrm{i} \in Z_{n^s}$, a plaintext is represented as a non-negative integer such that $\mathrm{i} < n$ and $r \in Z_n^*$ a random value. The ciphertext is defined as $E(\mathrm{i}, r) = (1+n)^{\mathrm{i}} r^n \bmod n^2$
- **Decryption** Given a ciphertext $c = E(\mathrm{i}, r)$ and $\mathrm{d} = 0 \bmod \lambda$, to retrieve the message m compute: $m = \mathrm{d}^{-1} \frac{(c^{\mathrm{d}}-1)}{n} \bmod n$.

3.1.3 Hashing

A cryptographic hash function is a function that takes a variable-size input and returns a fixed-sized output. Ideally, the hash functions can easily compute the hash value for a given message and it should impossible to generate a message given the hash nor to modify it without changing the hash. It can be seen as a one way function that produces a unique result, that means, it should be infeasible to find two different messages with the same hash.

3.1.4 Digital Signatures

The signature scheme is a method for *authenticating* data, this is, verifying that the data was approved or created by a certain party (or set of parties). Public-key signature schemes are universally verifiable. Each user can efficiently produce his/her signature on any document, and every other user can efficiently verify that signature, and it is infeasible to produce signatures for messages that other users did not sign.

The main idea of digital signatures is to provide assurance to the receiver that the data received was actually sent by the sender, and it has not been modified by someone else.

The authentication process consists of two main steps: *i)* signing and *ii)* verification.

A digital signature scheme $\mathcal{DS} = (\mathcal{K}, Sign, VF)$ consists of three algorithms, as follows:

- *Key generation algorithm* $\mathcal{K}$ that returns a pair (pk, sk) of keys
- *signing* algorithm Sign takes the secret key sk and a message M to return a signature
- *verification* algorithm VF takes a public key pk, a message M, and a candidate signature σ to return true or false depending on whether the signature is correct or not.

RSA

The RSA signature scheme uses the same technique as described in 3.1.2 with slight modifications. In order to sign a document, the creator first produces a hash value of the message, then applies the *signing* algorithm to the hash, this returns the signature. To verify the signature, the receiver takes the signature, applies the *verification* algorithm on it and compares the output with the message's actual hash value.

The Key Generation is the same, but we replace the algorithms decrypt and encrypt by signature and verification respectively as shown:

- **Sign:** On input $x \in Z_n$ and $SK = (n, \mathrm{d})$ output $y = x^{\mathrm{d}} \bmod n$.
- **Verify:** On input $y \in Z_n$ and $PK = (n, \mathrm{e})$ calculate $x = y^{\mathrm{e}} \bmod n$.

3.2 More advanced Cryptographic Protocols

In this section we describe some protocols and methods to enable parties to run protocols securely, which includes jointly computing a function over their inputs while at the same time keeping these inputs private, sharing a secret among different participants, and computing a proof of knowledge of a secret.

3.2.1 Secret Sharing

Adi Shamir in [Sha79] introduced a method for distributing a secret among a group of participants. The secret is divided into parts, giving each participant its own unique part, where either some of the parts or all of them are needed in order to reconstruct the secret. The essential idea of Adi Shamir's threshold scheme is based on polynomial interpolation: First, choose $k-1$ coefficients $a_0 \dots a_{k-1}$ at random and let a_0 be the secret to share. Then the polynomial is build as $f(x) = a_0 + a_1 x + a_2 x^2 \dots a_{k-1} x^{k-1}$. Then every participant is given a point (a pair consisting of input to the polynomial and the corresponding output). Given a subset of k of these parts, it is possible to find the secret value a_0.

3.2.2 Zero Knowledge Proofs

Zero-Knowledge proofs, introduced by Goldwasser, Micali and Rackoff in 1985 [GMR85], are proofs that assure a certain fact and yet yield nothing beyond the validity of the assertion being proved. That is, a verifier obtaining such a proof only gains conviction in the validity of the assertion. The notion of zero-knowledge proofs is a (multi-round) randomized protocol for two parties, called verifier and prover, in which the prover wishes to convince the verifier of the validity of a given assertion, whereas no prover strategy may fool the verifier to accept false assertions.

3.2.3 Non-Interactive Zero Knowledge

This model introduced by Blum, Feldman and Micali in [BFM88] is a variant of Zero Knowledge Proofs in which no interaction between the prover and the verifier is necessary. Non-Interactive Zero Knowledge consist of three entities: a prover, a verifier, and a string. Both verifier and prover can read the string. The interaction consist of a single message sent from the prover to the verifier, who is then left with the final decision (whether to accept or not).

3.3 Network Security Tools

3.3.1 Transport Layer Security (TLS)

The first version of the Transport Layer Security was presented in [DA99] . It was designed to provide communication privacy over the Internet. It allows client / server applications to communicate in a way that is designed to prevent eavesdropping, tampering or message forgery. This protocol uses cryptographic protocols in order to provide the following properties:

- **Privacy:** The data is encrypted using symmetric cryptography. The keys for the encryption are uniquely generated for each connection and are negotiated using asymmetric cryptography.
- **Reliability:** In order to assure the content of the message, all the messages are authenticated providing integrity. With hash functions it is possible to detect accidental or intentional message changes.

TLS is a *high level* protocol, meaning that it specifies different algorithms to do the handshake, how to interpret the authentication certificates, the session protocol, etc. TLS provides interoperability between different technologies and extensibility allowing the use of new cryptographic blocks. Currently it is the standard for Internet communication, electronic commerce and secure transactions.

In next chapter we introduce a general electronic voting design, presenting the entities, participants and procedures which are involved in an e-voting scheme.

Chapter 4

e-Voting Design

For the design, we formalize the entities involved in the voting protocol based on the generic scheme presented earlier.

4.1 Entities

1. **Registrars:** Denoted by $\mathcal{R} = R_1, R_2, \ldots, R_{nR}$, this is a set of nR entities responsible for setting up the election, selecting the authorities and the electoral roll, configuring the candidate slate $\mathcal{X}$ and verifying the election key generation.
2. **Authorities:** Denoted by $\mathcal{T} = T_1, T_2, \ldots, T_{nT}$, this is a set of nT authorities responsible for holding the election key (each one holds be a share) to calculate the sum of the encrypted votes (tally).
3. **Voters:** Denoted by $\mathcal{V} = V_1, V_2, \ldots, V_{nV}$, this is a set of n_V voters for the election administrated by $\mathcal{R}$. Each voter V_{i} has his own credential in order to authenticate himself.
4. **Servers:** A set of servers, responsible for receiving the votes and creating the ballots, verifying them and creating the proofs to enforce the honesty of voters. For each set of servers, every interested party controls at least one of them, to prevent collusion. These sets are:
   (a) Encryption Servers: Denoted by $\mathcal{E} = E_1, E_2, \ldots, E_{nE}$, a set of nE servers is responsible to receive plaintext votes and return a ballot with the respective proof that the vote is well formed.
   (b) Verification Servers: Denoted by $\mathcal{I} = I_1, I_2, \ldots, I_{nI}$, a set of nI servers is responsible to verify the correctness of the proofs and the signatures.
   (c) Re-encryption Servers: Denoted by $\mathcal{S} = S_1, S_2, \ldots, S_{nS}$, a set of nS servers is responsible to re-encrypt the ciphertext generated by one of $\mathcal{E}$ with the corresponding proof that the vote is well formed.
   (d) Re-encryption Verification Servers: Denoted by $\mathcal{C} = C_1, C_2, \ldots, C_{nC}$, a set of nC servers is responsible to verify the re-encryption process.

(e) Bulletin Board: Denoted by $\mathcal{BB} = B_1, B_2, \ldots, B_{nB}$, a set of nB servers which act as a piece of universally accessible memory to which all players have appendive-write access. In this memory any member of $\mathcal{V}$ can write data to $\mathcal{BB}$ but cannot overwrite or erase the existing data. This is used to publish all the ballots cast by voters.

4.2 Cryptographic primitives

Our e-voting platform uses two asymmetric schemes: $\mathcal{AE}_{\mathcal{A}}$ based on RSA and $\mathcal{AE}_{\mathcal{H}}$ with homomorphic properties.

Both schemes consists in three algorithms:

- $Keygen_{\mathcal{AE}}(k)$: On input k as security parameter, $Keygen$ outputs a Secret/Public Key pair.
- $Encrypt_{\mathcal{AE}}(pk, m)$: On input of the message m and the public-key PK, it outputs the ciphertext c.
- $Decrypt_{\mathcal{AE}}(sk, c)$: On input of the ciphertext c and the secret-key SK, it outputs the original message m.

We also use a symmetric encryption scheme $\mathcal{SE}$ to encrypt the secret key of $\mathcal{AE}_{\mathcal{A}}$. $\mathcal{SE}$ consists of two algorithms:

- $Encrypt_{\mathcal{SE}}(sk, m)$: Using an admissible secret key sk, convert the plaintext message m into ciphertext c.
- $Decrypt_{\mathcal{SE}}(sk, c)$: Taking the secret key sk, convert the ciphertext c into the original message m.

4.3 Setup

First, one of the registrars, say $R_1 \in \mathcal{R}$ creates the election and configures the basic parameters like name of the election, description, start date, finish date and selects the size of the encryption key. Also R_1 configures the slate candidate $\mathcal{C}$ with L candidates. The slate candidate contains all the questions and options. Then R_1 selects the list of voters $\mathcal{V}$ which can participate in the election and assign the authorities $\mathcal{T}$. Finally R_1 publishes the bulletin board $\mathcal{BB}$.

At the end of the Setup process, the platform creates the keys for $\mathcal{T}$ by invoking the Key generation algorithm.

4.4 Key generation

The key generation is an algorithm to create the keys involved in the election. Several keys are used in order to ensure the ballot privacy of voters and the secure distribution of keys. The intuition is as follows:

In order to maintain the ballot privacy, the platform creates one secret key sk for the symmetric encryption scheme $\mathcal{SE}$ and two asymmetric pairs of keys: One pair (SK_H, PK_H) to encrypt the preference using a homomorphic encryption scheme $\mathcal{CS}_{\mathcal{H}}$ and the other pair (SK_A, PK_A) using an asymmetric encryption scheme $\mathcal{AE}_{\mathcal{A}}$ to encrypt an optional text in the vote. In order to preserve privacy of the optional text, the key SK_A is encrypted using sk and afterwards sk is encrypted using PK_H. SK_H is distributed among the authorities $\mathcal{T}$ using a secret sharing scheme. So, there is no way to open the optional encrypted texts until the election is finished and all SK_H shares were combined.

Formally, the algorithm performs the following steps:

1. Generate a symmetric secret key SK_{SE} admissible for the message space of $\mathcal{CS}_{\mathcal{H}}$.
2. Run the $Keygen_{\mathcal{AE}}$ algorithm for $\mathcal{CS}_{\mathcal{A}}$ to create the key pair (SK_A, PK_A).
3. Run the $Keygen_{\mathcal{AE}}$ algorithm for $\mathcal{CS}_{\mathcal{H}}$ to create the key pair (SK_H, PK_H).
4. Using $\mathcal{SE}$, encrypt SK_A with SK_{SE} as secret key. $C_{SE} = Encrypt_{\mathcal{SE}}(SK_{SE}, SK_A)$.
5. Using $\mathcal{CS}_{\mathcal{H}}$ encrypt SK_{SE} with PK_H as public key. $C_{CSH} = Encrypt_{\mathcal{AE}}(PK_H, SK_{SE)}$.
6. Using a Secret Sharing scheme, distribute SK_H among the set of authorities $\mathcal{T}$. Let $S = s_1, s_2, \ldots, s_{nT}$ be the shares of SK_H.
7. Output $(PK_H, PK_A, S, C_{SE}, C_{CSH})$.

4.5 Share distribution

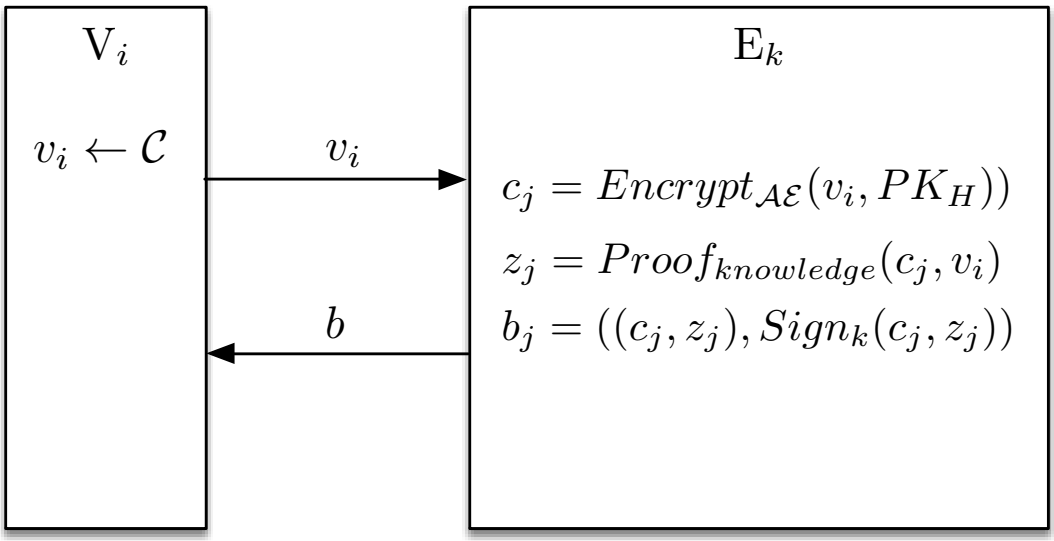
Each authority receives his share by email as an attachment in our implementation). If T_{i} has a PGP key pair, the distribution can be completed using his public key to enforce the privacy of the share. Otherwise, the Key generation process can be done offline, and the shares are distributed using usb flash drives.

4.6 Vote preparation

Voting selection takes place before authenticating the voter meaning that no credential is submitted by the voter during the vote selection process. This technique is aimed to preserve ballot privacy. An unauthenticated voter V_{i} sends his preference v_{i} to an E_k Encryption server controlled by one of the interested parties.

E_k computes a set b of ballots, such that, $b_j = ((c_j, z_j), Sign_k(c_j, z_j))$ for $j = 1$ to n, where $c_j = Encrypt_{\mathcal{AE}}(v_{\mathrm{i}}, PK_H)$ n different encryptions of v_{i}. z_j is a zero knowledge proof

Figure 4.1: Vote preparation



of knowledge with respect to the encrypted message c_j. After the computation, E_k signs the ballot in order to verify the authenticity of the E_k. The complete process is shown in Figure 4.1.

4.7 Cut and Choose

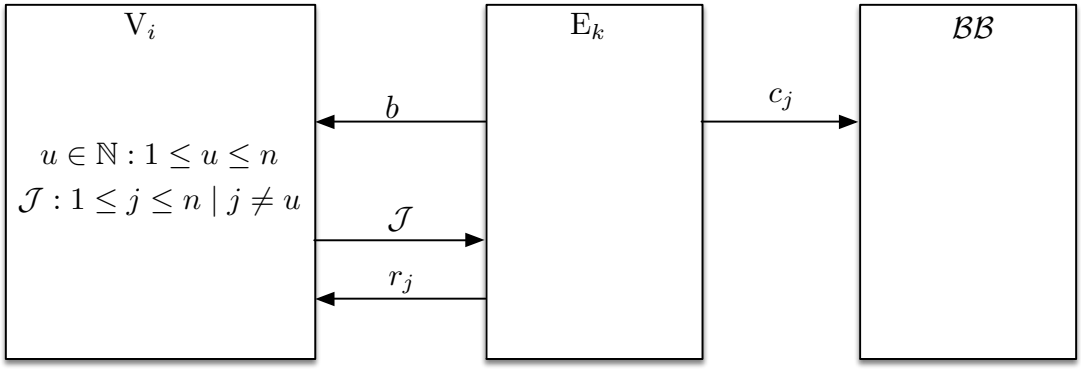
The idea in the Cut-and-choose phase is that, after the voter E_k returns the set of ballots b, V_i can verify if all encryptions were correct requesting from E_k the randomness used to encrypt some of the b_j, $1 \leq j \leq n$ in order to send it to an external verifier. This protocol can be performed several times until V_i is convinced that the encryption is in fact correct (with high probability).

Formally:

V_i receives the ballot set b, chooses a random u, $1 \leq u \leq n$ and sends it back to E_k all the indices j, $1 \leq j \leq n \mid j \neq u$. E_k returns each randomness r_j used to compute c_j and publishes each c_j on the bulletin board to avoid that those b_j can be used later.

After having completed the cut-and-choose phase, V_i obtains a set of encrypted votes b, each signed with their respective zero knowledge proof of knowledge, and the randomness r_j for b_j, $1 \leq j \leq n \mid j \neq u$.

Figure 4.2: Cut and choose

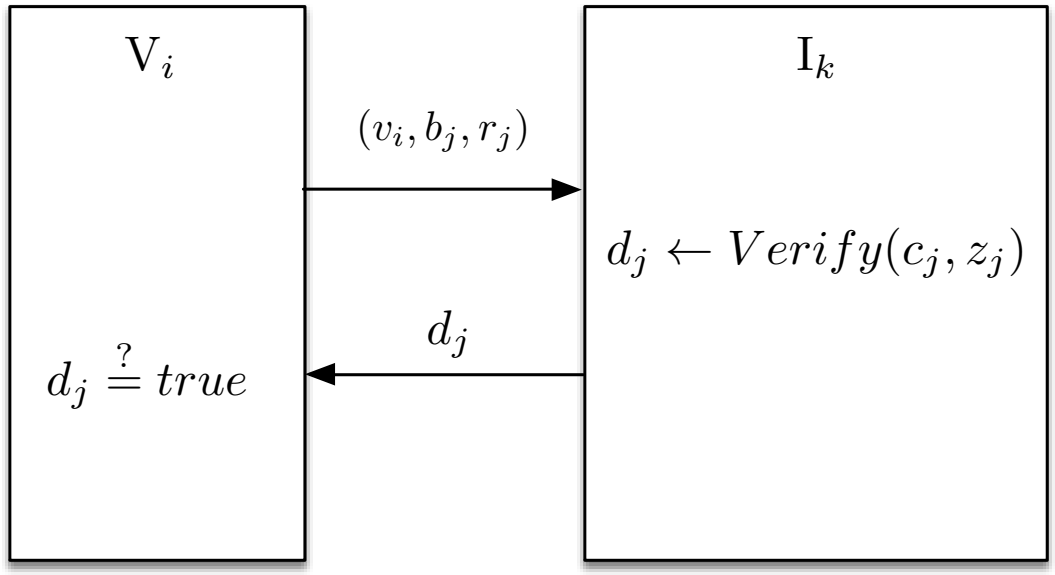


4.8 Vote verification

V_i might start a protocol in order to verify that the ciphertext is encrypted properly and that the vote was actually encrypted by an authentic $E_k \in \mathcal{E}$. After having received b and r_j for b_j, $1 \leq j \leq n \mid j \neq u$, V_i can verify that the vote is well formed and that the encryption is in fact his preference (w.h.p.).

First, V_i sends all b_j v_i and r_j for $1 \leq j \leq n \mid j \neq u$ to $I_k \in \mathcal{I}$. I_k first verifies if the signature of each b_j is correct using E_k's public key. Then he checks if the vote is well formed by encrypting the vote using the given randomness. This means I_k checks if $Encrypt_{\mathcal{AE}}(v_\text{i}, PK_H) \stackrel{?}{=} c_j$ using r_j.

Figure 4.3: Vote verification

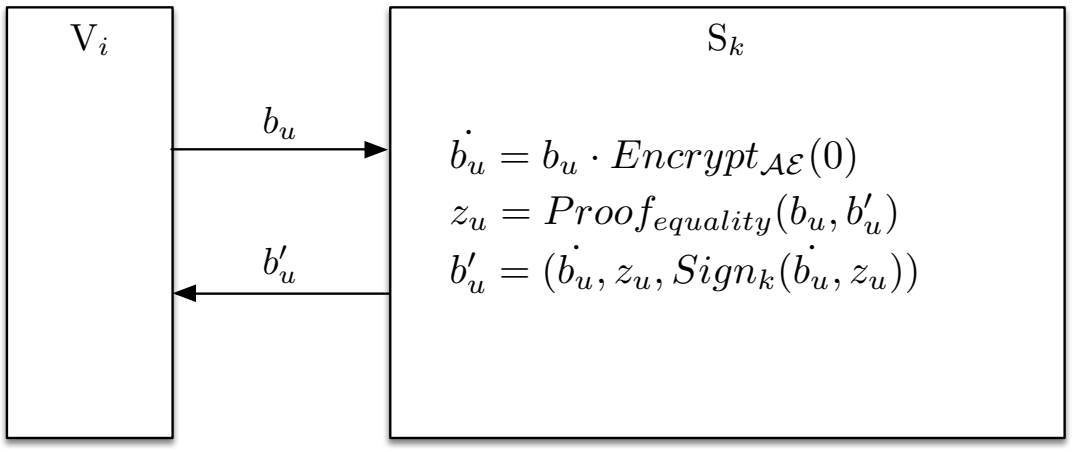


4.9 Re-encryption

After the Vote verification, V_i submits at least one of the ballots b_u that V_i did not request for randomness in the Cut-and-choose phase. All these ballots must be re-encrypted in order to preserve privacy of V_i.

V_i sends b_u to S_k, who computes the re-encryption $\dot{b_u} = b_u \cdot Encrypt_{\mathcal{AE}}(0)$, a zero knowledge proof of the equality of the plaintexts $z_u = Proof_{equality}(b_u, b'_u)$. Finally, S_k signs $(\dot{b_u}, z_u)$ and sends it back to V_i.

Figure 4.4: Re-encryption



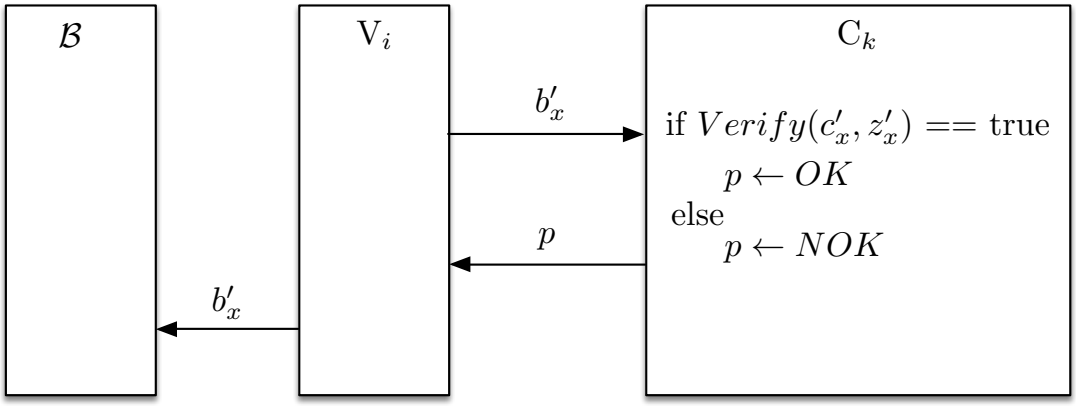
4.10 Voting

Before casting the ballot, V_i verifies that the re-encryption phase was performed correctly. V_i sends the ballot $b'_x = (\dot{b_u}, b_u, z_u)$ to C_k, and C_k verifies the signature of the ballot and that the re-encryption is correct.

Once V_i completed the last verification process, V_i submits b'_x, s.t. E_k *does not reveal the randomness*. After having checked that the signature is valid, the ballot is posted to the bulletin board $\mathcal{BB}$.

The re-encryption is an important process to preserve the secrecy of the vote, otherwise the encryption servers might know all votes since they encrypt the plaintext. Based on the assumption that the re-encryption and encryption servers are not under the control of the same adversary the secrecy of the ballot is preserved.

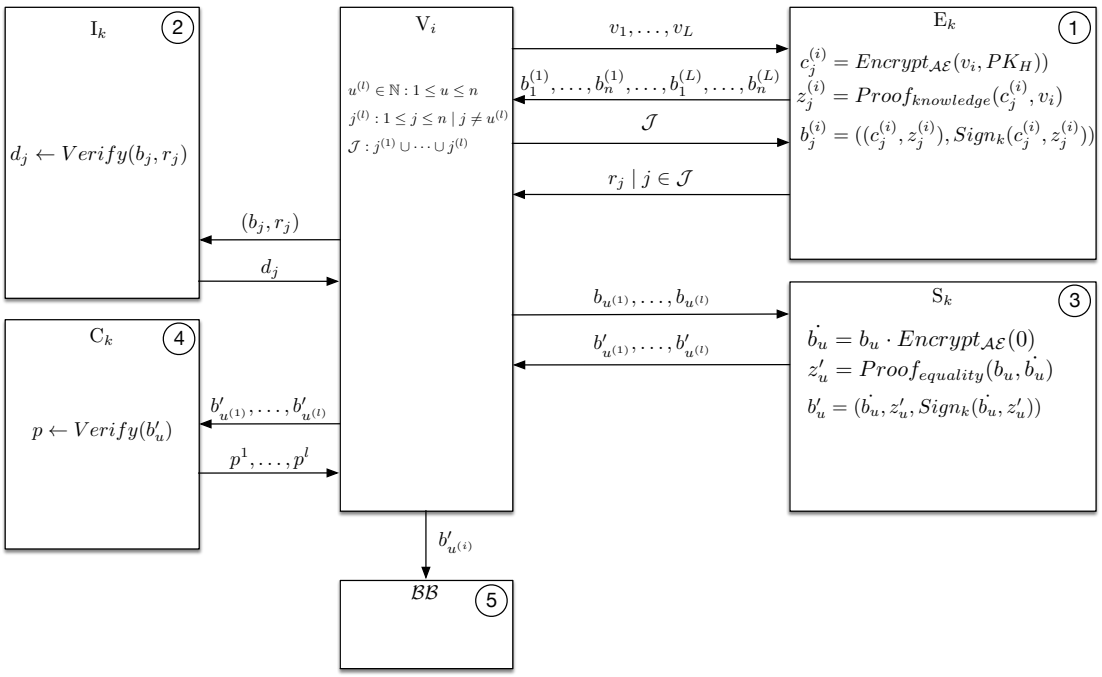
Figure 4.5: Voting



4.11 Complete voting process

In Figure 4.6 we present the full picture of the voting process, with all entities interacting.

Figure 4.6: Voting



4.12 Tallying

Using the homomorphic characteristics of the encryption scheme, it is possible to compute the sum of the all encrypted votes without decrypting them. Once the election is closed, the authorities sum all votes and create a new encrypted value which is the sum of all votes.

Anyone can access the bulletin board and calculate the sum by himself in order to verify that it is the same value that the authority computed and published.

4.13 Decryption

Once the sum was computed, a subset of size t, $nT/2+1 \leq t \leq nT$ of $\mathcal{T}$ is needed to decrypt the tally. On our platform, each authority uploads his keys into the system and performs a partial decryption. After all required authorities completed the partial decryption, all the pieces are combined in order to publish the result.

4.14 Characteristics

In our remote electronic voting design we achieve some characteristics and properties desirable for all e-voting platforms. We can ensure that the vote was counted in the final tally. Since all ballots are public and downloadable by any user, anyone can verify that the encrypted tally is correct and valid as well as that their vote was included, using the homomorphic properties. The platform does not give any receipt in order to prove the content of the ballot.

We provide a mark-in option that allows the voter to write something in the ballot. In order to do so, the text on the ballot is encrypted by using the secret key PK_A of the asymmetric encryption scheme $\mathcal{AE}_{\mathcal{A}}$. This option has been used in Chile to vote and express a protest or comment on something at the same time. By marking the vote, if the ballot is properly marked (i.e., the text is not over the vote line) the vote is valid and counted. Marking the ballot is not allowed in other countries because it can be used as a proof of coercing a voter. In our design, all the mark-in options are removed from the ballots before performing the sum. As soon as all SK_H shares are combined, the symmetric key sk is decrypted allowing the authorities to decrypt SK_A to decrypt all marked votes. In the current implementation we do provide privacy for the mark-in content, this could be easily added by running the mark-in content through a mix-net.

Using digital signatures it is possible to avoid ballot injection from the voters. Any ballot issued by anyone other than the authorities can not be inserted into the bulletin board, since the impossibility to forge a valid digital signature of any authority. All these characteristics imply non-repudiation, preventing any voter that cast a ballot to say that he did not.

Chapter 5

Implementation

5.1 Concrete Instantiation

In this work, we use some cryptographic blocks as we showed in 4.2. The concrete instantiation of the symmetric encryption scheme $\mathcal{SE}$ is an AES block cipher in CBC mode operation with PKCS5 Padding. For the asymmetric scheme $\mathcal{AE}_{\mathcal{A}}$ we use the RSA encryption scheme in ECB operation with PKCS1 Padding. Both schemes are very common in literature and the details can be studied in [Dwo01].

For the asymmetric homomorphic encryption scheme $\mathcal{SE}_{\mathcal{H}}$ we use Paillier's Cryptosystem. [Pai99]. Also we introduce in our work the optimizations and the threshold variant presented by Damgård *et al.* [DJN10]. One of the most important blocks for our work is the distribution of the secret key in order to prevent any collusion of authorities. The intuition is to distribute the election secret key among all authorities. A smaller subset of at least t such that $t \leq nT$ of them is required in order to decrypt the tally efficiently while less than t authorities obtain no useful information about the tally.

Paillier's threshold version is based on Victor's Shoup work presented in [Sho00]. Damgård's version is sightly similar with some differences, in particular the election of the secret key in order to not leak any information.

The formal description of the cryptosystem is as follows:

- **KeyGen** First generate two primes p and q with $p = 2p'+1$ and $q = 2q'+1$ such that p' and q' are different than p and q respectively. Set $n = pq$, $m = p'q'$ and pick d to satisfy $\mathrm{d} = 0 \bmod m$ and $\mathrm{d} = 1 \bmod n$. Then define the polynomial $f(X) = \sum_{\mathrm{i}=0}^{k-1} a_{\mathrm{i}} X^{\mathrm{i}} \bmod nm$, pick a_{i} (for $0 < \mathrm{i} < k$) as random values from $\{0, ..., n * m - 1\}$ and $a_0 = \mathrm{d}$. The idea is to recover a_0 with t authorities. Each authority will be $s_{\mathrm{i}} = f(\mathrm{i})$ for $1 \leq \mathrm{i} \leq l$. In order to verify the decryption, a public value v, which is a generator of a cyclic group of squares in $Z^*_{n^2}$, is published. For each decryption server a key is generated $v_{\mathrm{i}} = v^{\Delta s_{\mathrm{i}}} \bmod n^2$, where $\Delta = l!$

- **Encryption** Given $\mathrm{i} \in Z_{n^s}$ a plaintext is represented as a non-negative integer such that $\mathrm{i} < n$ and $r \in Z_n^*$ a random value. The ciphertext is defined as $E(\mathrm{i}, r) = (1+n)^{\mathrm{i}} r^n \bmod n^2$.
- **Share decryption** The i'th authority computes $c_\mathrm{i} = c^{2\Delta} s_\mathrm{i}$, where $\Delta = l!$ and c is the ciphertext. Along with the share, a zero-knowledge proof is computed such that $log_{c^4} = log_v(v_\mathrm{i})$ to convince that indeed she raised c to her secret exponent s_i.
- **Share combining** After having collected k shares (with their correct proof) it is possible to combine them by: $c' = \prod_{\mathrm{i} \in S} c_\mathrm{i}^{2\lambda_{0,\mathrm{i}}^S} \bmod n^2$, where $\lambda_{0,\mathrm{i}}^S = \Delta \prod_{\mathrm{i}' \in S \mathrm{i}} \frac{-\mathrm{i}'}{\mathrm{i}-\mathrm{i}'} \in Z$. The value of c' will have the form $c' = c^{4\Delta^2 f(0)} = c^{4\Delta^2 \mathrm{d}} = (1+n)^{4\Delta^2 M} \bmod n^2$, where M is the plaintext.

In section 4.6 we mention some zero-knowledge proofs to ensure that the computation has been done properly. Since our proofs are based on the work by Damgård *et al.* in [DJN10] (some of them are taken verbatim from there) we note that the following protocols are not zero-knowledge, but only honest verifier zero-knowledge. These protocols can be converted into Non Interactive Zero Knowledge using the Fiat-Shamir transformation this implies that we cannot obtain zero-knowledge, but are able to obtain security in the random oracle model and satisfy special soundness. All the proofs for the following protocols can be found in [DJN10].

Protocol for n^s th powers. Input: n, u Private Input for P: $v \in Z*_n$, such that $u = E(0, v)$.

- P chooses $r \xleftarrow{\$} Z*_n$ such that $u = E(0, v)$.
- V chooses e a random number of t-bit, and sends e to P.
- P sends $z = rv^{\mathrm{e}} \bmod n$ to V. V checks that u, a, z are relatively prime to n and that $E(0, z) = au^{\mathrm{e}} \bmod n^{s+1}$ and accepts if and only if this is the case.

Using the protocol for n^s th powers it is possible to build an efficient proof that an encryption contains one of two given values, without revealing which one it is, given M a honest verifier simulator for the n^s power protocol:

Protocol 1-out-2 n^s th power. Input: n, u_1, u_2 Private Input for P: $v1$, such that $u_1 = E(0, v_1)$.

- P chooses r_1 at random in $Z*_n$. He invokes M on input n, u_2 to get a_2, e_2, z_2. He sends $a_1 = (E(0, r_1), a2)$ to V.
- V chooses s a random number of t-bits, and sends s to P.
- P computes $\mathrm{e}_1 = s - \mathrm{e}_2 \bmod 2^t$ and $z_1 = r_1 v_1^{\mathrm{e}1} \bmod n$. He then sends $(\mathrm{e}_1, z_1, \mathrm{e}_2, z_2)$ to V.
- V checks that $s = \mathrm{e}_1 + \mathrm{e}_2 \bmod 2^t$, $E(0, z_1) = a_1 u_1^{\mathrm{e}_1} \bmod n^{s+1}$, $E(0, z_2) = a_2 u_2^{\mathrm{e}_2} \bmod n^{s+1}$, and $u_1, u_2, a_1, a_2, z_1, z_2$ are relatively prime to n.

Protocol Multiplication-mod-n^s. Input: $n, g, \mathrm{e}_a, \mathrm{e}_b, \mathrm{e}_c$ Private Input for P: a, b, c, r_a, r_b, r_c, such that $ab = c \bmod n$ and $\mathrm{e}_a = E(a, r_a)$, $\mathrm{e}_\mathrm{d} = E(b, r_b), \mathrm{e}_c = E(c, r_c)$.

- P chooses random values $\mathrm{d} \in Z_{n^s}, r_{\mathrm{d}}, r_{\mathrm{d}b} \in Z*_n$ and sends the encryptions $\mathrm{e}_{\mathrm{d}} = E(\mathrm{d}, r_{\mathrm{d}}), \mathrm{e}_{\mathrm{d}b} = E(\mathrm{d}b, r_{\mathrm{d}b})$ to V.
- V chooses e a random number of t-bits, and sends it to P.
- P opens the encryption $\mathrm{e}_a^{\mathrm{e}} r_{\mathrm{d}} \bmod n$. Finally, P opens the encryption $\mathrm{e}_b^f (\mathrm{e}_{\mathrm{d}b} \mathrm{e}_c^{\mathrm{e}})^{-1} = E(0, r_b^f (r_{\mathrm{d}b} r_c^{\mathrm{e}})^{-1} \mathrm{mod} n)$ by sending $z_2 = r_b^r (r_{\mathrm{d}b} r_c^{\mathrm{e}})^{-1} \mathrm{mod} n$.
- V verifies that the opening of encryptions in the previous step were correct, and, that all values sent by P are relative prime to n.

For the zero-knowledge proof z_j mentioned in section 4.6 we use the *smaller vote size variant* proposed by Damgård *et al.* in [DJN10]. To compute the proof correctly, the election must hold the following relation: $L \cdot log_2(M) < 2\dot{(}k-1)$ where k is a security parameter, $L = 2^{l+1}$ is the number of candidates for some k, and M is the number of voters, which is true in many realistic situations.

Using a proper encoding of the vote, consider a binary representation such that $j = b_0 2^0 + b_1 2^1 + \cdots + b_l 2^l$, V_{i} preference is for candidate j, each vote is defined to be an encryption of the number M^j. Clearly, the encryption is either 1 or a power of M. Using this, it is possible to construct the algorithm for producing the desired proof (P denotes the prover).

Also, it is possible to use the Proof of knowledge of an encrypted message presented in [BFP+01] by Baudron *et al.*

1. P computes encryptions $\mathrm{e}_0, \ldots, \mathrm{e}_l$ of $(M^{2^0})^{b_0}, \ldots, (M^{2^l})^{b_l}$ for each $\mathrm{i} = 0 \ldots l$. P also computes $Proof_P(\mathrm{e}_{\mathrm{i}}/(1+n)$ or $\mathrm{e}_{\mathrm{i}}/(1+n))^{M^{2^{\mathrm{i}}}}$ is an encryption of 0.
2. Let $F_{\mathrm{i}} = (M^{2^0})^{b_0}, \ldots, (M^{2^{\mathrm{i}}})^{b_{\mathrm{i}}}$, for $\mathrm{i} = 1 \ldots l$. P computes an encryption f_{i} of F_{i} for $\mathrm{i} = 1 \ldots l$. We set $f_0 = \mathrm{e}_0$. Now for $\mathrm{i} = 1 \ldots l$, P computes:

$$Proof_P(\text{Plaintext corr. to } f_{\mathrm{i}-1}, \mathrm{e}_{\mathrm{i}}, f_{\mathrm{i}} \text{ satisfy } F_{\mathrm{i}-1} \cdot (M^{2^{\mathrm{i}}})^{b_{\mathrm{i}}} = F_{\mathrm{i}} \bmod n^2) \tag{5.1}$$

For more details we refer to [DJN10].

5.2 System Architecture

Our e-voting platform achieves the desired requirements using to new technologies that we developed. In particularly our system relies on cloud computing as fundamental principle. Cloud Computing refers to technical approaches with the term, as well as to a business model in which core computing and software capabilities are outsourced to third companies like Amazon EC2, Microsoft Azure and Rack-Space among others.

The availability and characteristics of cloud computing is more than a single application hosted in a web server over a company network. It encloses multiple servers, multiple networks and multiple technologies, accessible from anywhere using an Internet connection. Using this

technology, companies can reduce the time in batch processing, since, nowadays, using 1000 servers for one hour costs no more than using one server for 1000 hours.

From a hardware provisioning point of view, there are four relevant aspects [ASZ+10]:

- The appearance of infinite computing resources available on demand, quick enough to follow load surges, and eliminating the planing of provisioning.
- Elimination of up-front commitment by cloud users; all users can start with small resources and increase it depending on their needs.
- The ability to pay for the use of computing resources on a short-term basis and to release them as needed.
- The opportunity to raise an entire infrastructure in a few minutes without deep understanding of all technical details.

Cloud computing was a fundamental part of this work and it was considered in the implementation design and decisions. We trade off some usability issues with privacy properties using realistic assumptions to relax both.

5.2.1 Setup

The Setup shown in section 4.3 describes an election initialization by the election administrator, the creation and configuration is carried out in an intuitive three-step web form screen: First set, the election name, start and finish date, questions and options.

After setting the desired parameters, the second step is to specify the voters who are eligible to vote. The web form requires some personal information to individualize each voter, like name, e-mail address, ID number and an optional secret password. This information is gathered to verify the identity of the voter. The set of all these voters is called electoral roll or roaster.

Finally, the election administrator adds the personal information for the election authorities to generate all the keys. To start the Key generation algorithm described in 4.4 it is possible to compute it online or offline depending on the desired confidence level for the election.

- **Online:** All keys are generated in one server which is used only for this purpose. The generation and distribution occurs on an external server without an administrator intervention. All shares are send to the corresponding authorities by e-mail.
- **Offline:** Each administrator downloads a small piece of software to generate the keys on their own computer. All shares are stored locally and are distributed via physical media.

After all shares have been created and delivered, a configuration file is created and distributed to the encryption servers to create the voting form.

5.2.2 Scale on demand

The vote selection web form is implemented as a front end of the Encryption Servers $\mathcal{E}$. These servers are exposed to the Internet and anyone can reach them. These servers render the vote with all preferences in blank for any anonymous voter. Voters submit their preference to the backend application using TLS and the backend will output the ballot as shown in 4.6.

Each election may be completely different from any other. This e-voting platform is targeted to social organizations, that may show unexpected behavior. If the election is about a national topic, it might have a high number of voters trying to access the platform. If only a small group of people are trying to decide their Union's next president, the interest will probably be limited.

This application may receive various amounts of requests in a particular timeframe. The need of servers processing the data may increase or decrease depending on the interest of people possibly from different parts of the world. Using the Amazon's Elastic Cloud Computing (EC2) we observe the behavior of each server and add or remove resources in a fine grain fashion (one server at a time) and with a lead time of minutes.

Real world estimates of average server utilization in data centers range from 5% to 20% [RCPS08][Sie08] and peak workload exceeds the average by factors of 2 to 10. In order to always be available, the provision must be at least the expected peak. Thus, resources are idle at non peak times. It is possible to identify two scaling behaviors, that increase or decrease the computational capacity of the application by either changing the number of servers (horizontal scaling) or changing the size of the servers (vertical scaling).

In this work, we use horizontal scaling and increase the number of Encrypt servers depending on the CPU consumption (as soon as 60% of the load is exceeded, a new instance is activated, while getting below 40% will remove one). We also create a *Network Consumption threshold*, which also scale the number of servers depending on the Bandwidth utilization.

5.2.3 High Availability

Before the rise of social networks, social groups were small and communication was personal. Now it only takes seconds for a webpage link to be spread on social media and gain big popularity. This can happen when the topic of an extremely sensitive election is currently popular, with the result that a significant portion of users – potentially hundreds of thousands of people – visit the webpage within a few hours, causing a Distributed Denial-of-Service (DDoS) effect. A DDoS is an interruption or suspension of services due to the overconsumption of resources. It can be unintentional or intentionally motivated by various reasons.

Using Netty, an optimized web server, it is possible to serve up to $\sim$ 40000 request per second on a Amazon EC2 large Instance [Tec13]. For all servers, we have same system configuration: A Netty web server is used in addition to an elastic platform to increase the number of servers on demand due to its high performance.

5.2.4 Fast and Reliable Bulletin Board

Every ballot that is submitted to the Bulletin Board passes through a verification process. First it checks if the signatures are valid, then if the proof of knowledge is valid. If any of these verifications fail, the ballot is discarded, otherwise the value is marked as valid and it is persistently stored in a database. Storage has always been a bottle neck because the speed of a hard disk is slower than RAM (nowadays it is an order of magnitude comparing SATA3 with DDR3).

In order to increase the bulletin board performance, we divide the bulletin board into two pieces: ballot verification and persistent storage. Decoupling components by implementing a fast queue as front end bulletin board removes the bottle neck and increases the throughput.

Each voter casts his ballot into this fast queue (push). An unsynchronized second process will take (pop) the ballots one after the other from the queue, verify them and store them in a distributed database.

The queue is reliable and stored redundantly and scalably allowing an unlimited number of reads and writes at any time.

5.2.5 Distributed Database

As persistence storage we use a distributed database across different locations to increase reliability. Using a distributed database in a cloud enables scaling it at the same time that the application is scaling. If for any reason one election extensively gets more intense people's interest, the database will scale as does the entire application.

We implement a Multi-Zone deployment, in order to endure critical workloads with high availability and automated fail-over from a primary database to a synchronously replicated secondary database in the case of failure.

Chapter 6

Trust Assumptions

Our e-voting platform relies on certain assumptions in order to create a practical electronic voting system. Taking into consideration the specific segment which is involved in our platform, these assumptions will hold in most cases.

6.1 Cloud computing provider

We trust in the cloud computing provider, we assume that they keep the uptime that they describe in their service level agreement (SLA), and that they do not cut off the entire election.

Also we trust that all the software running on their platform is malware-free. All the code that we upload into the different servers assures integrity, precluding accidental and intentional changes in the files.

If there is an adversary in the Cloud computing provider, he can view all code (binary mainly) as well the data stored, but he cannot modify it without forging signatures.

6.2 Voting clients

Our platform is 100% online, and the voters cast the ballots using a standard internet connection over TLS. In most of the cases the voter client (operating system and/or browser) is clean. This application is very specific and changes from one election to the next. We trust that the client is clean at least from tailor-made malware. Nevertheless there are easy countermeasures for this: use another browser or a bootable clean operating system.

This trust assumption does not require that voters trust in one software implementation that we provide. Voters may even choose which client to trust. In particular, if one voter wants be completely anonymous, he can even connect using a Tor network [DMS04].

6.3 Private channel

All voters use a private channel implemented over TLS. We trust the correctness of the certificate verification chain and that the browser does not have any adversary behavior accepting fake certificates as valid. An adversary can observe all network traffic, however all traffic is encrypted and the adversary can learn nothing just from observing. We assume that the adversary cannot modify any message without breaking the integrity.

6.4 Cryptographic assumptions

We rely on the typical cryptographic assumptions, such as RSA, decisional composite residuosity, and hash functions as a random oracle.

6.5 Mutually distrustful parties

In current democracies, there are at least two parties involved in elections. These parties are different and both are represented in each election. We assume that different parties involved will not collude to break the ballot privacy nor act as a unique adversary. We trust that at least one party is honest, as well as the registrars (responsible for the setup of the election). We define three different domains with n players in each domain. The first domain is the Encryption Servers $\mathcal{E}$, the second is Verification Servers $\mathcal{I}$ and Re-Encryption Servers $\mathcal{S}$, and the third one is the Re-encryption servers $\mathcal{C}$. In our trust assumption, we assume that only one domain can be under total control of the adversary.

6.6 Key Generation

In our system the key generation is done in a single computer which we assume that is uncorrupted. In practice this computer is offline and the generation process is done in a ceremony with physically present authorities. It is possible to remove this assumption by running a fully distributed key generation algorithm.

Chapter 7

Evaluation

Our implementation is functional alpha-level prototype. We tested our platform in the mainly used browsers (Internet Explorer, Chrome, Firefox and Safari) and in the most used operating systems (Windows, OS X, Linux) at this time. We combined existing web programming techniques and cryptographic protocols to provide an electronic voting system suitable for cloud computing.

In our first prototype, we model our e-voting platform with one server by entity, discarding the verification and re-encryption servers. We also discard the cut-and-choose implementation. We use the Java Virtual Machine (JVM v7) as Virtual Machine, running bytecode generated by the Scala 2.10 compiler. The source code is mainly written in Scala, with the exception of some libraries implemented in Java. In this version, we use a threat model in which an attacker does not have access to any of the servers. In particular, we completely trust the Encrypt Server, encrypting the actual voter preference and signing the ballot in order to prevent spurious votes generated by a malicious voter.

Encryption servers were configured in the Amazon Cloud Computing platform as an array to prevent any kind of distributed denial of service. After the key generation, each share was distributed to the corresponding authorities using a USB flash drive.

We show the experience and results for a real case e-voting scenario. We concretely implemented our e-voting platform for an election where Chileans - demanding their right to vote from abroad - where able to vote for the presidential elections in Chile in a non-binding (symbolic) election.

7.1 Context

For the occasion of the General Elections in Chile in November 2013, a campaign was launched encouraging Chileans living abroad to stand up for their voting rights abroad. In this context, the three organizations *Voto Ciudadano, Fundación Democracia y Desarrollo* and *Inria Chile* held a symbolic and electronic presidential election where Chileans living anywhere in

the world other than Chile were able to participate. Through this gesture, they expressed themselves through their right to vote. The platform was open between November 10th and November 17th 2013.

We took into consideration that every Chilean has one ID number *RUN (Rol Único Nacional)*, and that all identity documents (ID card and passport) have only one serial number, which is valid if the document is neither expired nor blocked. Using the public infrastructure of the national registry service *Registro Civil* and the national election service *Servicio Electoral* we validate the information provided by the voters. We validate that the voter has a valid national identity document and has the right to vote.

In order to prevent ballots casted from Chile, we filtered the voters by IP address using a GeoIP webservice. Depending on the origin of the IP we displayed different sites. For Chilean IPs we encouraged to the voter to go directly their polling station. For all the other IPs we showed the voting site. Since it is possible to spoof the IP Address using a proxy or VPN server, we constantly checked if the IPs were listed on public proxy servers and removed these ballots from the ballot-box.

7.2 Results

In the general election, none of the candidates held more than 50% of votes, and a second round was held on December 15th. After the success of the first initiative, a symbolic electronic ballotage also was realized.

In the first election, the platform received 12,418 ballots from 110 countries, receiving an average 67 ballots per hour, having a peak of voters in the last hours. We deployed all the servers in the Amazon Cloud Computing Infrastructure using Amazon Linux as operating system. For each server (Encrypt server, Ballot-box and Result server) we used a different virtual machine; each one replicated four times with automatic scaling and load balance in case of high demand or denial-of-service attacks. The Encrypt server is the most critical server because performing the cryptographic calculations requires a lot of CPU with a unpredictable behavior. In our case, the CPU load was under 5% almost all the time, as well as the one of the ballot box server[1]. The casting rate was 1.11 ballots per minute, which means that the server was idle most of the time.

7.3 First Election

In table 7.1 we show the top 7 ballots received by country.

Almost all the voting advertisement was by social networks and Chilean abroad organizations. Once the election became popular, it also was published in newspapers and television.

[1]In practice, ballot-box is *much* CPU consumer than encrypter.

Country	Ballots cast
UNITED STATES	2,180
SPAIN	1,360
ARGENTINA	1,171
CANADA	779
GERMANY	770
FRANCE	769
SWEDEN	676

Table 7.1: Ballots received by countries in first election

The most active communities were in Europe, in particular in Spain, Germany, Sweden and France.

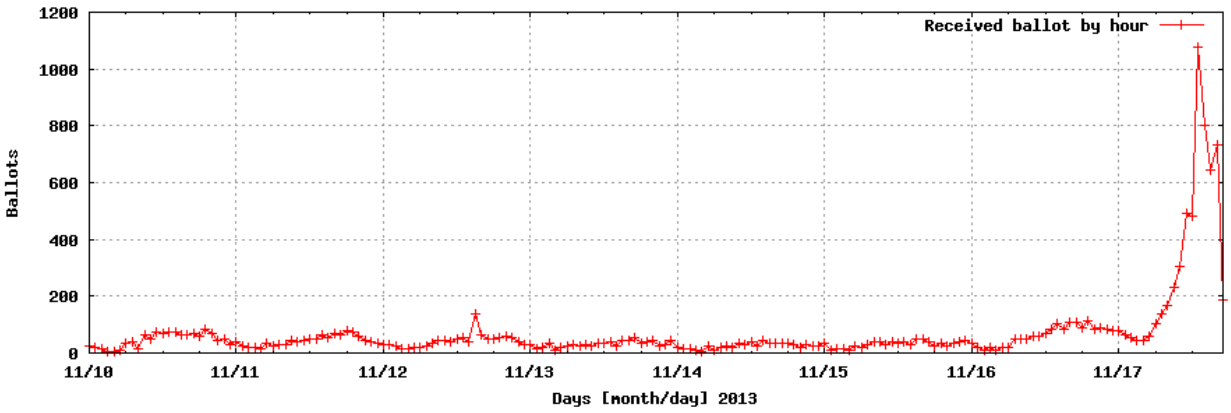


Figure 7.1: Average number of ballots received per hour for each day that the election was open.

In Figure 7.1 we present the ballots received per hour during all the days that the election was open. Voters cast their ballots systematically each day. Since we received ballots from 110 countries with different timezones, the server was not loaded at any hour except at the end. Close to the end of the election, the servers received a higher number of ballots, partly because of the publicity inviting people to be part of this election.

It was possible to vote in the election between the 10th and the 17th of November. During these days, 12,486 votes were cast, of which 12424 were valid (ballots cast from a foreign IP address, with voters using a valid identity document and enrolled to vote). In figure 7.2 we present a compact version of the average received ballots. There was a peak of 180 ballots by hour cast between 13:00 and 14:00 consistent with the last call to vote using a emailing platform. The final results for each candidate are summarized in Table 7.2.

Figure 7.3 illustrate the proportion of votes for each candidate. It is possible to see that there were not a absolute majority, but a strong tendency for one candidate. In the current Chilean legislation, the president must achieve an absolute majority in the general election, and since we tried to adjust the most possible to the legislation, a second non-binding election was held with the two first majorities achieved in the general election.

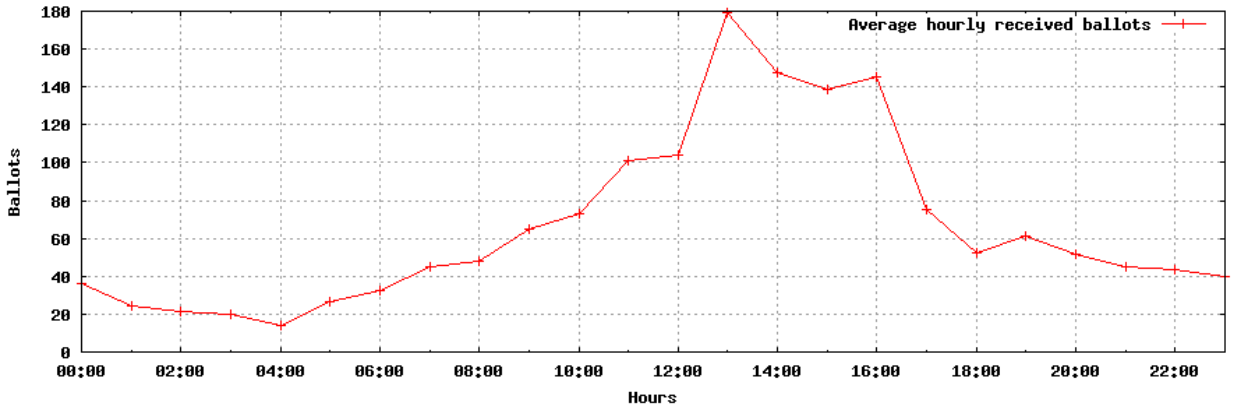


Figure 7.2: Average hourly received ballots.

Candidate	Votes
MICHELLE BACHELET	4,249
MARCEL CLAUDE	1,951
MARCO ENRIQUEZ-OMINAMI	1,736
EVELYN MATTHEI	1,489
ALFREDO SFEIR	1,318
FRANCO PARISI	704
ROXANA MIRANDA	704
RICARDO ISRAEL	63
TOMAS JOCELYN-HOLT	33

Table 7.2: Election Results

In this election, the social organization *Marca tu Voto* started a big campaign to initiate a constituent assembly to reform the Chilean constitution. Part of this initiative marked their ballot with "AC" to quantify the magnitude of the support. We followed this, and implemented an anonymous write-in option to mark the ballot. $1,360$ voters used this option and wrote something, $1,289$ voters marked "AC".

7.4 Second round (runoff voting)

The second round was held between the 9th and the 15th of December using the same infrastructure and configuration. In this election, $5,699$ ballots were cast. In Table 7.3 we show the top 7 ballots received by country.

The voter participation in the second round was less than half of the general election (45,6%). Figures 7.4 7.5 show that on average the ballots cast were less than in the general election. In Figure 7.4, there are two peaks of around 100 ballots by hour and one peak in the last hours of the election achieving 250 ballots by hour. This can be explained by the advertisement campaign using an emailing platform to send a kind reminder to voters. The result was $4,553$ votes for Michelle Bachelet and $1,145$ votes for Evelyn Matthei.

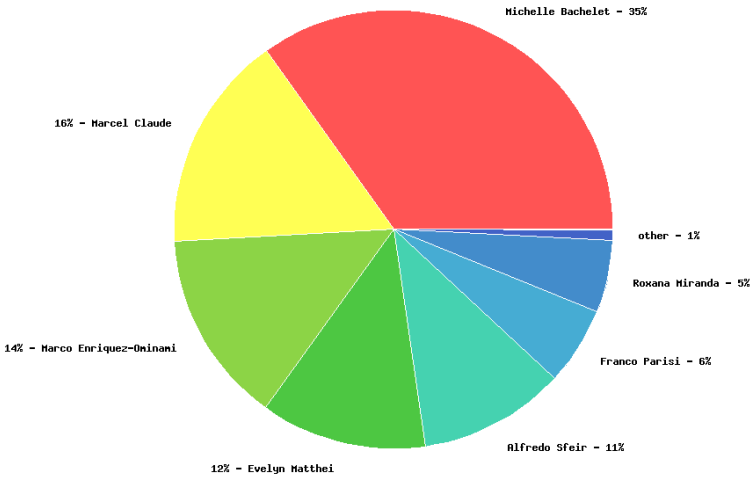


Figure 7.3: Election results

Country	Ballots cast
UNITED STATES	974
SPAIN	633
ARGENTINA	422
FRANCE	405
CANADA	365
SWEDEN	342
GERMANY	284

Table 7.3: Ballots received by countries in second round

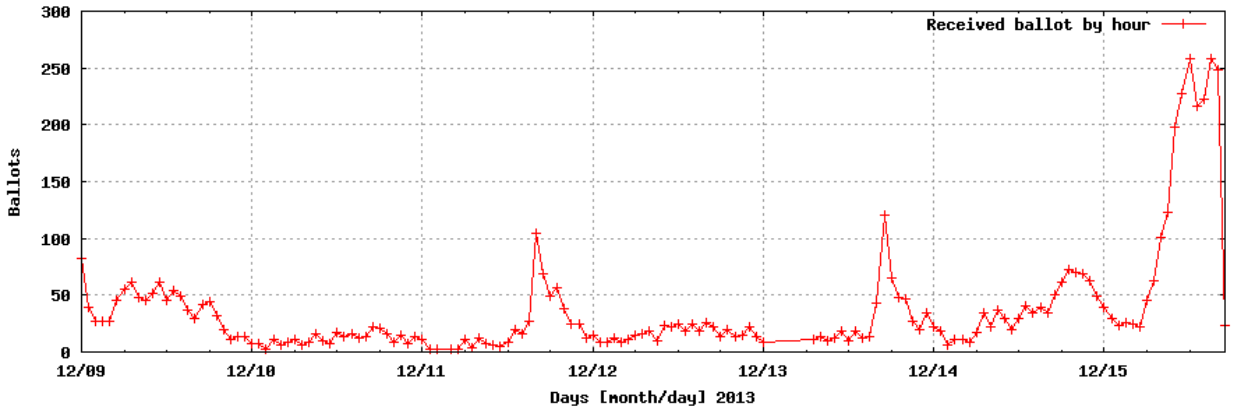


Figure 7.4: Average received ballots per hour

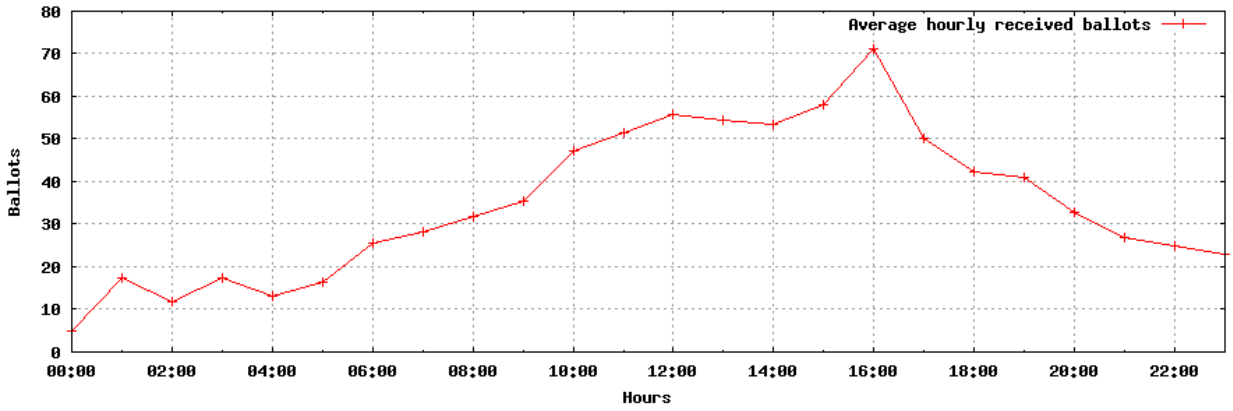


Figure 7.5: Average hourly received ballots

Chapter 8

Conclusion

In this work we present a remote electronic voting design, an implementation and a concrete election using cloud computing capabilities. Although our cryptographic tools are no new, provide a practical implementation as well as an example as to where this kind of remote electronic voting system can be used.

We implemented a concrete electronic voting system which takes the heavy task of performing cryptographic computations off the client. We rather move the encryption, decryption, proof generation and validation to an elastic cloud computing platform. This is a change in the actual paradigm, since most of the current remote electronic voting implementations try to encrypt on the client side. Instead of making the user the high computation, we transfer the trust to well-known entities and organizations such as universities, political parties, national services, etc. Voters can use any infrastructure based on the reputation or service level agreement.

We explore this idea of moving all heavy computation from the client side while preserving privacy and verifiability. The latter two important characteristics can be found in any remote voting systems. As soon as there are several trusted parties, the probability of changing a voter's preference is reduced.

In our work we focus on the client experience, trying to reduce entry barriers and keep the platform easy to use. To cast a ballot, voters use a standard web browser, using TLS in order to create a secure channel between the server and the voter browser. The current tendency is to move from computers with high computational power to smartphones and portable devices. We believe that the work presented is a step into that direction.

Working with social organizations is different from small clubs or student governments. Social organizations mainly use social networks to communicate and spread their opinions or initiatives. This has a direct impact on the design of platforms because of the unpredictability of people's reactions. We can observe this in two practical experiences. A symbolic runoff voting showed different levels of interest in the different rounds: In the first round, $12,424$ votes were cast, meanwhile in the second round, only $5,699$ were cast, using the same advertisement campaigns and media coverage.

An interesting analysis is that the amount of votes of the second round $(5,699)$ is similar to the sum of votes of the two first round candidates Michelle Bachelet $(4,249)$ and Evelyn Matthei $(1,489)$. An explanation for this behavior could be that the voters that did not vote for these candidates in the first round did not transfer their votes to the candidates in the second round, but rather did not vote at all.

This work is a step towards new applications of cloud computing, in particular with elastic computing architecture. In the practical experiences presented, it was possible to observe different behaviors during the elections. In particular, the last hours were the most critical because of the high interest of people to participate. The Elastic computing alternative is a better option than trying to estimate the whole infrastructure for an unpredictable load peak. It helped us to achieve 100% uptime during the elections while reducing costs and maintenance time.

We provide practical implementations of multiple cryptographic protocols and algorithms with a strong background in to maintaining ballot privacy and correctness in one of the most important processes in a democracy – which is voting. We hope that our platform will generate further interest in creating participative applications.

Bibliography

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *In Proceedings of the 17th USENIX Security Symposium, (Security '08)*, 2008.

[Alc11] Soledad Alcaide. Movimiento 15-M: los ciudadanos exigen reconstruir la política, May 2011.

[And01] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.

[ASZ+10] Michael Armbrust, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, and Ariel Rabkin. A view of cloud computing, 2010.

[Bar11] Alexei Barrionuevo. With Kiss-Ins and Dances, Young Chileans Push for Reform, 2011.

[BBC11] BBC. Hundreds of Occupy Wall Street protesters arrested, 2011.

[BCP+11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *Proceeding ESORICS'11 Proceedings of the 16th European conference on Research in computer security*, pages 335–354, September 2011.

[Ben87] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, January 1987.

[Ben06] Josh Benaloh. Simple verifiable elections. In *EVT'06 Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop Pages 5-5*, page 5, August 2006.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. *Non-interactive zero-knowledge and its applications*. ACM, 1988.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, PODC '01, pages 274–283, New York, NY, USA, 2001. ACM.

[BnFL+12] Jonathan Ben-nun, Niko Farhi, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-shma, and Douglas Wikström. A New Implementation of a Dual (Paper and Cryptographic) Voting System. *Electronic Voting*, pages 315–329, 2012.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing - STOC '94*, pages 544–553, New York, New York, USA, May 1994. ACM Press.

[BWB05] Bruce Beckles, Von Welch, and Jim Basney. Mechanisms for increasing the usability of grid security. *Int. J. Hum.-Comput. Stud.*, 63(1-2):74–101, July 2005.

[CCM08] M.R. Clarkson, S. Chong, and A.C. Myers. Civitas: Toward a secure voting system. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 354 –368, may 2008.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Cha88] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. *Proceeding Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88*, pages 177–182, April 1988.

[Ciu14] Ciudadano Inteligente. Ciudadano inteligente, April 2014.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, January 2013.

[DA99] T Dierks and C Allen. The TLS Protocol Version 1.0, 1999.

[Del11] Manuel Delano. El 'invierno estudiantil' sacude Chile, 2011.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[DJN10] Ivan Damgård, Mads Jurik, and JesperBuus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[DKRD06] Stephanie Delaune, Steve Kremer, Mark D. Ryan, and Stéphanie Delaune. Coercion-resistance and receipt-freeness in electronic voting. In *In Proc. 19th Computer Security Foundations Workshop*, pages 28–42. Press, 2006.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.

[Dwo01] Morris Dworkin. Recommendation for Block Cipher Modes of Operation. Technical report, NIST, Washington, 2001.

[FCS] Kevin Fisher, Richard Carback, and Alan T. Sherman. Abstract punchscan: Introduction and system definition of a high-integrity election system.

[FIP01] Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES), 2001.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Proceeding ASIACRYPT '92 Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251, December 1992.

[Fun14] Fundación Iguales. Fundación iguales, April 2014.

[Gab11] Adam Gabbatt. Occupy Wall Street: protests and reaction, 2011.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion resistant end-to-end voting. In *Financial Cryptography*, pages 344–361, 2009.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[HS00a] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *Advances in Cryptology Proccedings of Eurocrypt 00*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556. Springer Berlin / Heidelberg, 2000.

[HS00b] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceeding EUROCRYPT'00 Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, pages 539–556, May 2000.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society - WPES '05*, page 61, New York, New York, USA, November 2005. ACM Press.

[Jur12] Jeffrey S. Juris. Reflections on #Occupy Everywhere: Social media, public space, and emerging logics of aggregation. *American Ethnologist*, 39(2):259–279, May 2012.

[KTV10] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-Based Definition of Coercion-Resistance and Its Applications. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 122–136. IEEE, July 2010.

[Man12] Essam Mansour. The role of social networking sites (SNSs) in the January 25th Revolution in Egypt. *Library Review*, 61(2):128–159, 2012.

[Mat12] Refugio Mata. #YoSoy132: Mexican Elections, Media, and Immigration, 2012.

[NR94] Valtteri Niemi and Ari Renvall. How to Prevent Buying of Votes in Computer Elections. In *Proceeding ASIACRYPT '96 Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*, pages 164–170, November 1994.

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *IFIP World Conference on IT Tools*, pages 21–30, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 25–35, April 1997.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, pages 223–238, 1999.

[Pas12] Stefano Passini. The Facebook and Twitter revolutions: Active participation in the 21st century. *Human Affairs*, 22(3):301–312, June 2012.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Proceeding EUROCRYPT '93 Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 248–259, January 1994.

[Rai11] Sarah Rainsford. Spain's 'Indignants' lead international protest day, 2011.

[RCPS08] Kash Rangan, Alan Cooke, Justin Post, and Nat Schindler. The Cloud Wars : 100 + billion at stake. *Analyst The*, (May):1–90, 2008.

[Rev14] Revolucion Democratica. Revolucion democratica, April 2014.

[RSA78] R. L. 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.

[Sca06] Virginia Scardamaglia. 600 mil "pingüinos" paran en las aulas, 2006.

[SFD+14] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security analysis of the Estonian Internet voting system. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*. ACM, November 2014.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sho00] Victor Shoup. Practical Threshold Signatures. In Bart Preneel, editor, *Advances in Cryptology, EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer Berlin Heidelberg, 2000.

[Sie08] Ludwig Siegele. Let it Rise: A Special Report on Corporate IT. *The Economist*, 25 Oct 200(October):8–11, 2008.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In *Proceeding EUROCRYPT'95 Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques*, pages 393–403, May 1995.

[SW12] Andrea Schmitz and Alexander Wolters. Political Protest in Central Asia. Technical Report April, Stiftung Wissenschaft und Politik, 2012.

[Tay11] Kate Taylor. Arab Spring really was social media revolution, 2011.

[Tec13] TechEmpower. Frameworks Round 2, 2013.

[Vot14] Voto Ciudadano. Voto ciudadano, April 2014.