Secure Electronic Voting Protocols

Helger Lipmaa, Cybernetica AS and University of Tartu (Estonia)

Contents

Keywords: Cryptography, electronic voting, homomorphic encryption, verifiable shuffle

Abstract

This chapter gives a summary of cryptographic e-voting protocols. We start with a motivation and an overview of some aspects of traditional elections. We then outline the security requirements of e-voting and present in detail two approaches to cryptographic e-voting, based on homomorphic encryption and mix networks. We finish the chapter with a list of open problems.

1 Introduction

During the last five or ten years, one of the global buzzwords (or rather, buzz-letters) has been "*e*", short from *electronic*, that signifies almost everything, connected with (inter)networking. The ubiquity of "e" is caused by the global penetration of the Internet, and already in many places of the world, by easier availability of Internet-based services, compared to the traditional services. Numerous e-processes are already taking place, starting from e-banking and ending with (in some countries) e-government. This had led to the situation where one wants to "e-ise" most of the processes and services that can be found in a modern society. After all, moving to the e-services helps one to cut down costs and save time. Additionally, it makes it possible for more and more people to become a part of the global society, and to benefit from its services.

Not surprisingly, also "e-ising" (nation-wide or local) elections promises to give measurable benefits. The very basic idea of the elections is to give every citizen of a country (or some other political unit) an equal right when deciding about the future of their country. To guarantee equal rights, it is essential to achieve a high voter turnout. As the most extreme case, look at an imaginary political system that has only two parties with election where "winner

takes all". If only 51% of the voters vote and the winning party collects 51% of the participating votes, the resulting government does not necessarily represent the majority of the citizens.

This is exactly where e-voting could help: it is at least one's hope that when voting is made more convenient, considerably more voters will turn out. And what could be more convenient than voting at your own home by using your own personal computer? Or your laptop, when travelling—or even your mobile phone when you do not have access to the Internet? Moreover, if votes are submitted electronically, vote counting could be almost instantaneous—in the contemporary world, quick vote counting is unfortunately an important issue. And last but not least, e-voting could make it cheaper and easier to organise elections.

But alas, convenience is not everything. Traditional voting booths have been designed to prevent vote coercion. How to prevent coercion when a user votes from home? How to ascertain that voter's computer functions correctly (no viruses, Trojan horses or keyboard sniffers)? How to ascertain that voting centre's computer functions correctly? (Denial of Service attacks, insider attacks, ...) Asking from a voter's—who may not know anything about how a computer or the Internet works—point of view, can one guarantee the correctness and robustness of the elections? And privacy? Can one protect the voter against coercers?

The answer is "partially, yes". Under some feasible *cryptographic* assumptions, privacy can be protected, although the voter still needs to trust his *voting platform*—e.g., a computer or a mobile phone—, and other pieces of hardware that are not under his or her own control. (But see also Sect. 8.) If, additionally, special hardware is used, one can design coercion-free elections. The use of secure cryptographic protocols together with fault-tolerant, well-organised, sufficiently duplicated and constantly verified voting infrastructure *might* also guarantee robustness. The necessary cryptography is already out there, together with a developing understanding of what are (at least some of) the specific requirements on the infrastructure and on the voters. Real understanding cannot come before people have gone through many trials and errors that result from electronic elections with significant, non-duplicated, and non-reputable outcome. It is our feeling that these real-life requirements will never be satisfied, even if some new breakthrough in cryptography makes some of the requirements obsolete.

Before going further, some warnings. E-voting means at least two quite different things: *Internet-voting*—voting over the Internet, as discussed above, by using a personal computer or mobile device with the possible help of minimal additional specialised hardware, or *kiosk-voting*—voting in some fixed location (like library, kiosk) by using special hardware; kiosk-voting also includes the current practice of some countries to use electronic or mechanical devices—like optical scans—in the voting stations. Kiosk-voting can be easier to organise—there is no question about the untrusted operating system on the voter's computer, for example—but it does not allow convenient "anywhere voting". (However, note that a diligent user can analyse software, running on his or her computer, while software, running in a kiosk is most likely going to be non-accessible.) Still, kiosk-voting might increase the voter turnout—and definitely decreases vote counting time. In this survey, by e-voting we will primarily mean Internet-voting. All the protocols that we will present can also be used in the case of kiosk-voting, although some of their features might be an overkill. On the other hand, even in the case of kiosk-voting, the current cryptographic solutions—even if used 100% correctly, which is rarely the case—are not yet completely satisfying.

In this chapter, we are going to present in detail several (under appropriate definitions) secure and (relatively) efficient cryptographic e-voting protocols. We will discuss the level of security achieved by the described protocols and also their efficiency. We will stop on the requirements on the infrastructure that seem to be necessary for the state of the art cryptographic e-voting protocols to fulfil their promise. Finally, we outline some important open questions and further research directions.

Notation. Let cands be the number of candidates, somehow enumerated by integers from 0 to $\mathsf{cands} - 1$. Let $V = (V_1, \ldots, V_{\mathsf{voters}})$ be the set of eligible voters. Let $V' \subseteq V$ be the set of eligible voters who turned up and voted. Let v_i be the vote as cast by V_i; we assume $v_i = \perp$ if V_i did not vote. Let μ be the concrete voting mechanism that is being used.

2 Voting: General Overview

During an election of e.g., national or local government, voters cast votes to a number of candidates. After the voting phase, the winning candidate(s) are computed from the set of votes. There are many game-theoretically sound (or

just validated by practice) *voting mechanisms* for the latter part. Next, we briefly summarise some of the well-known voting mechanisms (in all cases, the candidate with the most points wins). As in the rest of the paper we assume that there are voters voters and cands candidates. In all four cases, every voter outputs an ordered list of cands candidates.

Plurality: A candidate receives 1 point for every voter that ranks it first. (Here, it is sufficient for the voter to output only the top choice.)

Borda: For each voter, a candidate receives $\mathsf{cands} - 1$ points if it is the voter's top choice, $\mathsf{cands} - 2$ if it is the second choice, $\ldots$, 0 if it is the last.

Single Transferable Vote (STV): The winner determination process proceeds in rounds. In each round, a candidate's score is the number of voters that rank it highest among the remaining candidates, and the candidate with the lowest score drops out. The last remaining candidate wins. (A vote transfers from its top remaining candidate to the next highest remaining candidate when the former drops out.)

Maximin: A candidate's score in a pairwise election is the number of voters that prefer it over the opponent. A candidate's number of points is the lowest score it gets in any pairwise election.

If a mechanism μ is used, let $\mu(v_1, \ldots, v_{\mathsf{voters}})$ be the result of the election given votes v_i.

Exactly how the voting process is organised, depends largely on the individual country, and sometimes also on the individual county or even the village. However, an election tends to have at least the following phases, where the specifics of every phase may vary wildly:

Voter registration: All/most of the/some of the citizens are automatically registered as voters. The rest must register themselves as eligible voters.

Voting: During a few pre-announced days, every registered voter can cast his or her vote. At some a priori known time moment, the voting phase will be over. Voting period may depend on the individual tallier. (Thus, this model includes absentee voting.)

Tallying: After the end of voting phases, talliers count their tallies that are then mixed together to obtain the final result. (This phase depends heavily on the voting mechanism, the size of the country, etc.)

In practice, plurality and Borda elections are somewhat easier to organise than STV and Maximin since in them, only the total count of points for every candidate is needed for every candidate. This total count can just be incremented, as more and more votes from different voting stations become counted. This also means that when there are are many talliers—e.g., corresponding to different counties—, different sub-tallies can just be added up.

3 E-Voting: General Setting

Different countries implement elections in a different way, to comply and cope with their own traditions, definition of democracy, size of the country, etc. E-voting must take all such considerations into account and thus, just to make e-voting understandable to the voters, at least at first, e-voting must largely mimic conventional voting. In particular, an e-voting should have a registration phase, a voting phase and a tallying phase. Only later, when voters have become used to the e-voting, one could change the election process to suit better the specifics of e-voting.

Thus, we will think of e-voting as just a method to make voting more convenient, by enabling both the voters and the talliers to use technology to speed up their part of the process. Maybe later on, changes caused by e-voting will cause a revolution in the voting process—and thus, in the whole society. Currently, the change offered by e-voting is (or at least, in our opinion, should be) rather evolutionary.

How would (evolutionary) e-voting look like in an ideal world? First, the voters enter their votes to the *voting platform* (e.g., a computer). Then, the votes get transmitted to a central machine (the *tallying platform*) that computes the winner by using a fixed voting mechanism. Finally, the talliers output the name of the winner (or winners) with other auxiliary information that may be necessary (e.g., the number of votes of every candidate). In such an ideal world, all parts of the system function correctly. In particular, ideal e-voting has the next two important properties:

Correctness: The output of the elections is $\mu(v_1, \ldots, v_{\mathsf{voters}})$. That is, the election outputs a correct result, meaning that only the votes of legitimate voters count.

Privacy: During the election, nobody will gain new information about any of v_i-s—except what follows from $\mu(v_1, \ldots, v_{\mathsf{voters}})$ and their own private inputs. This includes at least the next subgoals: (a) Voter's preferences remain private. (b) Voting is coercion-free: even if you choose so, you are not able to later prove your vote. (c) Independence: voter should know his or her vote.

All mentioned subgoals correspond to the necessity of avoiding certain attacks. For example, imagine a simple voting protocol where every voter sends its encrypted vote to the tallier. Bob, a huge fan of a singer called Alice, just copies her encrypted vote and enters this to the voting platform. Or may be, Bob is able to manipulate the ciphertext so that his vote is the opposite of Alice's vote. This does not violate Alice's privacy (only under very special circumstances, Bob gains any information about Alice's vote!) but it creates undesirable situations where voters vote as their idol does—or as their hated one does not. This means that Bob must know his vote.

Next, we will give the definition of a secure electronic voting protocol. The definition is not fully formal, since in practice, it is not clear what is meant by "security." Moreover, some of the electronic voting protocols, presented in the following sections, do not satisfy the ideal security definition.

Assume that we have a fixed voting mechanism μ, like Plurality or Borda. Let Φ_μ be the function that, given votes of participating voters, computes some intermediate result that is necessary for finding the winner. For example, $\Phi_{\mathsf{Plurality}}(v_1, \ldots, v_{\mathsf{voters}})$ is usually a function that returns a vector $(w_1, \ldots, w_{\mathsf{cands}}, w_\perp)$, where $w_i = \sharp\{j : v_j = i\}$ is the number of voters V_j that voted for the ith candidate. It is possible to define $\Phi_{\mathsf{Plurality}}(v_1, \ldots, v_{\mathsf{voters}}) = w$, where w is the name of the winner. However, such solutions are not usually considered in the case of paper-ballot voting, since the privacy of losing candidates is usually hard to implement. The concrete definition depends on the voting traditions of an individual country. For example, if the number of seats in the parliament is proportional to the number of votes every party achieves, the full vector $(v_1, \ldots, v_{\mathsf{voters}})$ must be revealed.

Exactly like the conventional election, an e-voting protocol consists also of the registration phase, the voting phase and the vote counting phase. In the registration phase, the legitimate voters obtain the right to participate in e-voting. How this is done depends heavily on the traditions and technological infrastructure. For example, in some countries the voters may be able to register by using their ID-cards. This is largely a political issue, and we will just assume that legitimate voters will be able to vote, and obtain necessary information (like the public keys of the authorities) in an authenticated manner.

In the voting and tallying phase, we make a comparison with the "ideal world". In the ideal-world e-voting protocol, the trusted third party $\mathcal{T}$ keeps a database of votes. Every voter V_i casts a vote v_i that may be equal to $\perp$. The third party $\mathcal{T}$ stores the vote in her database. (A voter might be able to vote several times, but then only the result of the last vote counts.) After the end of the voting phase, $\mathcal{T}$ computes $\psi = \Phi_\mu(v_1, \ldots, v_{\mathsf{voters}})$. In the tallying phase, the value ψ is published. The tallier $\mathcal{A}$ finds the winner(s) of the election, based on ψ, by using rules, induced by the mechanism μ. (This part of the election is repeatable and verifiable by everybody.)

It is required that at the end of the protocol, the participants should have no information about the private inputs and outputs of their partners, except for what can be deduced from their own private inputs and outputs. In particular, V_i has no information about the value of v_j for $j \neq i$, and $\mathcal{A}$ has no information about the value of v_i for any i, except what they can deduce from their own private inputs and outputs. In practice, it usually means that it is required that the voting centre gets to know how many voters voted for every candidate, but not how did every single voter vote.

In an ideal world, exactly three types of attacks are possible [Gol04]: a party can (a) refuse to participate in the protocol; (b) substitute his or her private input to the trusted third party with a different value; or (c) abort the protocol prematurely. In our case, the attack (c) is irrelevant, since V_i has no output in the voting phase, and $\mathcal{T}$ has no output in the tallying phase. (Attack (c) models the case when the first party halts the protocol after receiving his private output but before the second party has enough information to compute her output.)

Therefore, in an ideal-world e-voting protocol, we cannot protect against a participant, who (a) refuses to participate in voting (*non-participation attack*) or (b) enters vote, different from her preference (may correspond to vote manipulation). No other attacks should be possible. Neither (a) nor (b) is traditionally considered an attack in the context of voting. The argument here is game-theoretic, and the solutions must be proposed by mechanism design (and politics!), instead of cryptography: namely, a non-manipulable mechanism (e.g., the algorithm with what the election

winner is determined from all the collected votes) must be designed so that answering against one's true preference (or non-participation) would not give more beneficial results to the respondent than the truthful answer.

On the other hand, as we stated, no other attacks should be allowed. This requirement is very strict, so we will explain why it is necessary in the voting context. Clearly, one must protect the privacy of voters: it is required that in democracy, one should be able to vote according to his or her true preferences. There are many cases where non-private voting could damage the interests of the individual voter (starting from a quarrel with his or her significant other, and ending with the possibility of getting discriminated by the new government, against whom one just voted).

It is also necessary to protect the privacy of $\mathcal{A}$, although the reason here is more subtle. Namely, if V_i obtains any additional information about ψ before the end of the elections, he or she might halt the protocol or change his or her vote. This might always happen since by a classical result of voting theory, all non-dictatorial voting mechanisms can be manipulated [Gib73, Sat73]. As an easy example, a voter can decide to vote for his or her second preference, when the first preference has no chance to win. (Halting the e-voting protocol while having no information on ψ is equivalent to the non-participation attack.) The third requirement on the protocol, of course, is that $\mathcal{A}$ either halts or receives $\Phi_\mu(v_1, \ldots, v_{\mathsf{voters}})$.

In a real-world implementation, we want to replace $\mathcal{T}$ by a cryptographic protocol $\Pi = (V_1, \ldots, V_{\mathsf{voters}}; \mathcal{A})$ between V_i and $\mathcal{A}$. This protocol $(V_1, \ldots, V_{\mathsf{voters}}; \mathcal{A})$ is assumed to be "indistinguishable" from the ideal-world protocol, that is, with a high probability, it must be secure against all attacks except (a) and (b). "Secure" means that the privacy of V_i (resp. $\mathcal{A}$) must be protected, if V_i (resp. $\mathcal{A}$) follows the protocol, and that $\mathcal{A}$ either halts, or receives the value $\Phi_\mu(v_1, \ldots, v_{\mathsf{voters}})$. Note that in particular this means that all messages between voters and $\mathcal{A}$ bust be authenticated by say using digital signatures.

Ideally, the security of the voters should be information-theoretical (that is, even an omnipotent adversary should not be able to violate the privacy of voters), while the security of tallier $\mathcal{A}$ can be computational (that is, a computationally bounded adversary should not be able to force $\mathcal{A}$ to accept an output that is not equal to $\Psi_\mu(v_1, \ldots, v_{\mathsf{voters}})$). This is since the voters, if they cheat, must do it online, while the adversary has all the eternity to violate voters's privacy. However, it is much easier to design e-voting with computational voter-security. In particular, all protocols that are described in the next sections provide only computational voter-privacy.

In a majority of existing secure e-voting protocols, every participant proves in zero-knowledge [GMR89] that he or she behaved correctly. (Sometimes, it is sufficient to have weaker guarantees, e.g., to present witness indistinguishable proofs.) Every voter must be able to verify the correctness of the zero-knowledge proofs, and thus can verify that his or her vote was counted with. In this case, one talks about *voter-verifiable (or voter-verified) electronic elections*. If the zero-knowledge correctness proofs are not only verifiable by the designated verifier but for everybody, including the casual observers, one talks about *universally verifiable electronic elections*. In practice, it is important that electronic (including both Internet and kiosk) elections were universally verifiable. Without universal verifiability, there is no hope of having any reliable "vote recounting" in the case of over-voting or under-voting, and no hope of correcting the errors in current kiosk-voting. See, e.g., [Ver04] for a high-profile campaign for universal verifiability.

Finally, note that the security requirements of e-voting schemes are different from the requirements of say electronic banking. One could assume that e-banking is at least to some extent reliable, since in the case of cheating, bank would get out of business. However, the sitting government will get out of business also when it loses the election, and moreover, has means to influence operators of e-voting systems. This is one of the reasons why universal verifiability is a must in the case of e-voting.

4 Cryptographic preliminaries

A public-key cryptosystem is a triple $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where Gen is the key generation algorithm that generates a private/public key pair $(\mathsf{sk}, \mathsf{pk})$, Enc is the encryption algorithm and Dec is the decryption algorithm. For a fixed Π and public key sk, we denote the corresponding plaintext space by $P = P(\Pi, \mathsf{pk})$, randomness space by $R = R(\Pi, \mathsf{pk})$ and ciphertext space by $C = C(\Pi, \mathsf{pk})$. Denote the encryption of a message $m \in P$ as $\mathsf{Enc}_{\mathsf{pk}}(m; r)$ where pk is the used public key and $r \in R$ is the used random coin. Throughout this paper, let κ denote the security parameter.

IND-CPA secure homomorphic cryptosystems. Define

$$\mathsf{Adv}^{\mathsf{pkcsem}}_{\Pi,\kappa}(A) := |\Pr[(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa), (m_0,m_1) \leftarrow A(1^\kappa,\mathsf{pk}), b \leftarrow_r \{0,1\}, r \leftarrow_r R : A(1^\kappa, m_0, m_1, \mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_b;r)) = b] - \frac{1}{2}| \ .$$

We say that Π is *IND-CPA secure* if $\mathsf{Adv}^{\mathsf{pkcsem}}_{\Pi,\kappa}(A)$ is negligible in κ for any probabilistic polynomial-time machine A. That is, Π is IND-CPA secure iff it is difficult for a polynomially bounded adversary to distinguish between random encryptions of two elements, chosen by herself.

Assume that the C (resp. P) is a group with group operation $\cdot$ (resp. $+$). Assume that R is a groupoid with groupoid operation $\circ$. Π is *homomorphic* when $\mathsf{Enc}_{\mathsf{pk}}(m_1;r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2;r_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1+m_2; r_1 \circ r_2)$ for any valid public key pk, plaintexts m_i, and random coins r_i.

The first well-known IND-CPA secure homomorphic cryptosystem ElGamal was proposed by ElGamal [El 84]. In the conventional ElGamal, one fixes two large primes p and q, s.t. $q \mid (p-1)$, and lets G_q be the unique subgroup of $\mathbb{Z}_p^*$ of order q. Let g be a generator of G_q. Private key is a random element $\mathsf{sk} \leftarrow_r \mathbb{Z}_q$. The corresponding public key is $h \leftarrow g^{\mathsf{sk}}$. Encryption is defined as $\mathsf{Enc}_{\mathsf{pk}}(m;r) := (g^r; mh^r)$. A ciphertext (c,d) can be decrypted by $m \leftarrow d/c^{\mathsf{sk}} = mh^r/g^{\mathsf{sk}r}$. Since $\mathsf{Enc}_{\mathsf{pk}}(m_1;r_1)\mathsf{Enc}_{\mathsf{pk}}(m_2;r_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1m_2; r_1+r_2)$, P is the multiplicative group $(G_q,\cdot)$. ElGamal is IND-CPA secure under the Decisional Diffie-Hellman assumption [TY98].

In several e-voting protocols, one needs an additively homomorphic cryptosystem, that is, where $P = (\mathbb{Z}_n,+)$ for some (possibly key-dependent) n. One can modify ElGamal to behave like an additively homomorphic cryptosystem by defining $\mathsf{Enc}_{\mathsf{pk}}(m;r) := (g^r; g^m h^r)$, but in this case decryption is feasible only when m is known to belong to some relatively small set (e.g., $m \in \{0,1\}$).

Paillier's cryptosystem Paillier [Pai99] is the first well-known IND-CPA secure additively homomorphic cryptosystem. Its IND-CPA security is based on the Decisional Composite Residuosity Assumption [Pai99]. Paillier's cryptosystem was extended by Damgård and Jurik to allow encryption of large messages [DJ01]. In the Damgård-Jurik cryptosystem DJ01, $n = pq$ is the public key, and its factorisation (p,q) is the secret key. One sets $\mathsf{Enc}_{\mathsf{pk}}(m;r) := (1+n)^m r^{n^s} \mod n^{s+1}$, where $s \geq 1$ can be freely chosen after n is generated. Here, $m \in \mathbb{Z}_{n^s}$ and $r \leftarrow_r \mathbb{Z}_n^*$. (In practice, $r \leftarrow_r \mathbb{Z}_n$ suffices.) For decrypting, one first computes $\mathsf{Enc}_{\mathsf{pk}}(m;r)^{(p-1)(q-1)} = (1+n)^{m(p-1)(q-1)} r^{\varphi(n^s)} \mod n^{s+1} = (1+n)^{m(p-1)(q-1)} \mod n^{s+1}$ and recovers m from that by using an algorithm from [DJ01].

Another similar cryptosystem, DJ03, was proposed by Damgård and Jurik in [DJ03]. DJ03 is slower and has longer ciphertexts than DJ01, and its IND-CPA security is based on both the Decisional Diffie-Hellman and the Decisional Composite Residuosity Assumptions being true. On the other hand, it has a simpler threshold version than DJ01.

Threshold homomorphic cryptosystems. The goal of a threshold cryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is to make it possible to share the private key among a set of receivers, so that only authorised sets of servers can decrypt messages. As always, Gen is the key generation algorithm, Enc is the encryption algorithm and Dec is the decryption algorithm. In the case of a threshold cryptosystem, the key is generated jointly by all participants so that everybody knows the public key pk, and all servers will have shares of the private key sk. Decryption is done by an authorised set of servers without explicitly reconstructing the private key. On the other hand, encryption algorithm is mostly used by outsiders who might not know that decryption is done in a threshold manner.

Next, we will describe the *threshold ElGamal Cryptosystem*, mostly because of its simplicity. A description of the more complicated threshold DJ01 and DJ03 cryptosystems can be found from [DJ01, DJ03]. Let Gen be a subgroup of $\mathbb{Z}_p^*$ of order q, where q and p are large primes. To generate a secret key $s \in \mathbb{Z}_p$, every server Server_j generates a share s_j as in Shamir's secret sharing scheme [Sha79]. That is, $s_j = f(j)$ for some polynomial f that is unknown to any single server. There exists exactly one polynomial f of degree k such that $s_j = f(j)$ for $j \in \{1,\ldots,k\}$. Server_j commits to her share s_j by publishing the value $h_j \leftarrow g^{s_j}$. As in Shamir's secret sharing scheme, the secret s is equal to $s = f(0)$.

By the Lagrange interpolation formula, given k points (x_i, y_i), s.t. $y_i = f(x_i)$, $i = 1,\ldots,k$, $f(x) = \sum_{i=1}^k y_i \prod_{j=1, j\neq i}^k \frac{x-x_j}{x_i-x_j} \pmod p$ (here, $x_j = j$ and $y_j = s_j$), and thus $s = f(0) = \sum_{i=1}^k c_i s_i \pmod p$, where $c_j := \prod_{j=1,j\neq i}^k \frac{-j}{i-j} \mod p$.

Therefore, g^s can be computed as $\prod_{j \in X} h_j^{c_j}$ from the public values h_j only, where X is any subset of k authorities. Then, $h = g^s$ is announced as the public key. No collection of $< k$ servers learns s, but note that s is only computationally hidden (w.r.t. the discrete logarithm problem).

To decrypt $(x, y) = (g^r, mh^r)$, the servers Server_j perform the following steps: (a) each Server_j broadcasts $w_j = x^{s_j}$, and proves in zero-knowledge that $\log_g h_j = \log_x w_j$; (b) let X be any subset of k authorities who passed the zero-knowledge proof. The plaintext can be recovered by X as $m' \leftarrow \frac{y}{\prod_{j \in X} w_j^{c_j}}$. Really, $w_j^{c_j} = x^{c_j s_j} = g^{r c_j s_j}$, thus $m' = mg^{rs} / \prod g^{r c_j s_j} = m$.

How to prove equality of discrete logarithms? Chaum and Pedersen [CP92] proposed the following protocol where A proves that $x = g^\mu \wedge y = h^\mu$ for some μ: (a) Prover generates a random $r \leftarrow_r \mathbb{Z}_q$, and sends $(a, b) \leftarrow (g^r, h^r)$ to verifier. (b) Verifier sends a random $e \leftarrow_r \{0,1\}^t$, $t \geq 80$, to prover. (c) Prover sends $z = r + \mu e \mod q$ to verifier. (d) Verifier checks that $g^z = ax^c$ and $h^z = by^c$.

Commitment schemes. A commitment scheme is a function $C : X \times R \to Y$ from the plaintext space X and random coin space R to the commitment space Y. A commitment scheme C is (a) *statistically hiding* if the commitment y = $C(m; r)$ leaks a statistically insignificant amount of information about the plaintext m and the coin r; and (b) *computationally binding* if given commitment $y = C(m; r)$ to some element r from the plaintext space, it is hard to find $m' \in P$, $m' \neq m$, and an r', s.t. $y = C(m'; r')$. For the best known commitment schemes (e.g., Pedersen's [Ped91]), the plaintext space is equal to $\mathbb{Z}_n$ for some n. Therefore, $C(m; r) = C(m + n; r)$ and therefore, such commitment schemes are not binding over the integers.

Fujisaki and Okamoto proposed an *integer commitment scheme* [FO99] that is binding over the integers. Their scheme was later improved by Damgård and Fujisaki [DF02]. The Damgård-Fujisaki integer commitment scheme is computationally binding and statistically hiding, given some reasonable cryptographic assumptions. Moreover, one can construct a very efficient honest-verifier statistical zero-knowledge (HVSZK) argument of knowledge that given three commitments c_1, c_2 and c_3, the prover knows such integers μ_1 and μ_2 and corresponding random coins ρ_1, ρ_2 and ρ_3, that $c_1 = C(\mu_1; \rho_1)$, $c_2 = C(\mu_2; \rho_2)$ and $c_3 = C(\mu_1\mu_2; \rho_3)$.

The homomorphic property of integer commitment schemes together with the efficient HVSZK argument of knowledge for the multiplicative relation can be used to construct efficient HVSZK arguments of knowledge of type $c_1 = C(\mu_1; \rho_1) \wedge \cdots \wedge c_n = C(\mu_n; \rho_n) \wedge c_{i+1} = C(\mu_{i+1}; \rho_{i+1}) \wedge \mu_{i+1} = p(\mu_1, \ldots, \mu_n)$, where p is an arbitrary polynomial $p \in \mathbb{Z}[X_1, \ldots, X_n]$. Lipmaa [Lip03] proposed a uniform methodology for constructing efficient HVSZK arguments of knowledge for a relatively large class $\mathcal{D}$ of languages; it is conjectured but not known that $\mathcal{D} = \mathcal{NP}$ [Lip03]. Given a statistically hiding and computationally binding integer commitment scheme with efficient HVSZK arguments of knowledge for additive and multiplicative relations, one can argue in HVSZK that she knows an auxiliary (suitably chosen) witness ω, such that $\mathfrak{R}_S(\mu; \omega) = 0$, where $\mathfrak{R}_S$ is the representing polynomial of S [Mat93, Lip03]. In particular, this results in an sub quadratic-length *Diophantine argument system* for all languages from the class L_2 of bounded arithmetic.

The *range argument* $y = C(\mu; \rho) \wedge \mu \in [L, H]$ has a HVSZK argument of knowledge with linear length $\Theta(|\mu|) \cdot \kappa$ [Lip03]. It is based on the famous theorem of Lagrange that every nonnegative integer μ can be represented as $\omega_1^2 + \cdots + \omega_4^2$ for some integers ω_i. The corresponding values ω_i can be computed efficiently [Lip03]. (See [Gro04] for a slight refinement.)

Efficient RAIE. In the next, we need an honest-verifier zero-knowledge proof of knowledge that the prover has encrypted a value of form voters^j where $j \in [0, \mathsf{cands} - 1]$ for some publicly-known constants voters and cands. This is called a *range argument in exponents* (RAIE). The currently most efficient honest-verifier computational zero-knowledge (HVCZK) RAIE was proposed by Lipmaa, Asokan and Niemi [LAN02]. The resulting RAIE has communication $\Theta(\max(k, \mathsf{cands} \log \mathsf{voters}) \cdot \log \mathsf{cands}) = \Theta(\mathsf{cands} \cdot \log \mathsf{voters} \cdot \log \mathsf{cands})$. For RAIE, one can use another function $a^{[\![i]\!]}$ instead of the exponentiation a^i [Lip03]. The function $a^{[\![i]\!]}$ is defined as follows. All nonnegative integral solutions (x, y) of the equation $x^2 - axy - y^2 = 1$ are either equal to $(a^{[\![i+1]\!]}, a^{[\![i]\!]})$ or $(a^{[\![i]\!]}, a^{[\![i+1]\!]})$, $i \geq 0$, where $a^{[\![i]\!]}$ can be computed by using the next recurrent identities [Mat93]: $a^{[\![0]\!]} := 0$, $a^{[\![1]\!]} := 1$, and $a^{[\![i+2]\!]} := aa^{[\![i+1]\!]} - a^{[\![i]\!]}$ for

$i \geq 0$. Thus, $\{a^{[\![i]\!]}\}_{i\in\mathbb{N}}$ is a Lucas sequence. When $a > 2$ and $i > 0$ then $(a-1)^i \leq a^{[\![i+1]\!]} \leq a^i$. Also, $a^{[\![i]\!]}$ can be computed almost as efficiently as a^i.

Bulletin board. A bulletin board is a public broadcast channel with memory where a players write information that any party can read. See, e.g. [CGS97].

5 Homomorphic E-Voting Schemes

Assume that the election uses the Plurality mechanism. (Implementing other mechanisms by using the next approach is also possible, although much more cumbersome.) Then, secure e-voting can be achieved as follows by using a secure homomorphic threshold cryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ [CGS97, DJ01]: Let a be the upper limit to the number of voters. Let $\tau \geq 1$. There is $2\tau + 1$ servers that share a public key pk and a private key sk so that everybody can encrypt a message by using pk, but only $\geq \tau + 1$ collaborating servers can jointly decrypt the ciphertext. Assume $v_i \in [0, \mathsf{cands} - 1]$ corresponds to the preferred candidate. The ith voter encrypts voters^{v_i} by using the key pk and sends it to the servers. The servers collect all ciphertexts and return receipts to the voters. The list of all encrypted votes is written on the bulletin board. After the end of the election, the servers multiply all ciphertexts, getting $y = \mathsf{Enc}_{\mathsf{pk}}(\sum_i \mathsf{voters}^{v_i}) = \mathsf{Enc}_{\mathsf{pk}}(\sum \alpha_j \mathsf{voters}^j)$, where α_j is the number of voters who voted for the candidate j. Thus, the servers can jointly decrypt y, and then compute the coefficients α_j. The value y together with the vector $(\alpha_1, \ldots, \alpha_{\mathsf{cands}})$ is published.

Next, we will look into the details of this generic *homomorphic e-voting* protocol.

Guaranteeing Correctness. To guarantee the correctness of this protocol, all voters must prove or argue in zero-knowledge that they encrypted a value of form a^j where $j \in [0, \mathsf{cands} - 1]$. This corresponds to the RAIE. The function $a^{[\![i]\!]}$ is exactly as suitable as a^i to be used as the encoding function that the voters use in the homomorphic e-voting scheme [Lip03]. Due to the fact that $(a^{[\![i]\!]})^2 - aa^{[\![i]\!]}a^{[\![i+1]\!]} - (a^{[\![i+1]\!]})^2 = 1$, there is a $\Theta(\mathsf{cands} \log \mathsf{voters})$-bit HVSZK argument of knowledge to prove that a voter voted correctly. Lipmaa, Asokan and Niemi [LAN02] proposed an alternative RAIE that is also based on the methodology from [Lip03]. Instead of the function voters^i (or $\mathsf{voters}^{[\![i]\!]}$), it uses the function b^i, where b is the least prime $b \geq \mathsf{voters}$. Since voters is fixed a priori and publicly known, b can be computed before the electronic voting starts. This RAIE is approximately as efficient as the RAIE based on the Lucas sequences: the arguer must argue that the committed value μ is such that $b^L \leq \mu$ and $\mu \mid b^H$. As later shown in [DGS02], one can simplify the argument even more by assuming that $b = p^2$ for a prime p: then one has to argue the knowledge of an ω, for which $(\omega \mid p^{H-L}) \wedge (\omega^2 p^{2L} = \mu)$. The RAIE is the single most communication-consuming sub-protocol of the homomorphic voting scheme. Therefore, the use of HVSZK arguments results in a $\Theta(\log \mathsf{cands})$-fold decrease of total communication.

Server's correctness can be verified by every voter by multiplying all the votes on the bulletin board, checking that their own votes are there, that the product is equal to y, and finally, that $\sum \alpha_j \mathsf{voters}^j$ is a correct decryption of y (by verifying another zero-knowledge proof).

Multi-candidate voting. The homomorphic e-voting scheme is especially efficient when used together with the additive variant of ElGamal. However, this is true only when $\mathsf{voters}^{\mathsf{cands}}$ is relatively small: the decryption results in $g^{\sum \alpha_j \mathsf{voters}^j}$, from which $\sum \alpha_j \mathsf{voters}^j \in [0, \mathsf{voters}^{\mathsf{cands}} - 1]$ can be found by solving the restricted discrete logarithm problem. The realistic value of voters is in $\{1, \ldots, 10^8\}$, depending on the elections. In the two-candidate case, when $\mathsf{cands} = 2$, and assuming that $\mathsf{voters} = 10^7$, $\mathsf{voters}^{\mathsf{cands}} \leq 10^{14} \leq 2^{47}$. Finding the corresponding discrete logarithm can be done in time $O(\sqrt{\mathsf{voters}^{\mathsf{cands}}}) \leq 2^{24}$, which is still realistic in most of the cases. However, for $\mathsf{cands} > 3$, we must look at alternatives to ElGamal.

The DJ01 cryptosystem can serve as a natural alternative. By using DJ01, the servers directly recover $\sum \alpha_j \mathsf{voters}^j$, and thus the costly discrete logarithm computation can be avoided. Moreover, values up to say $\mathsf{voters}^{\mathsf{cands}} \approx 2^{4096}$ (this corresponds to say $\mathsf{voters} \leq 10^8$ and $\mathsf{cands} \leq 150$) can be tolerated without significant

performance loss. On the other hand, the threshold DJ01 cryptosystem is slower and less convenient than the threshold ElGamal cryptosystem. Some compromise is offered by the DJ03 cryptosystem. However, at this moment, the choice of existing IND-CPA secure homomorphic cryptosystems is not completely satisfying.

The currently most efficient multi-candidate homomorphic voting protocol is described in [DGS02].

6 Verifiable Shuffle-Based E-Voting Schemes

In the verifiable shuffle-based approach (initiated by Chaum [Cha81]), every voter encrypts his or her vote v_i by using a public-key cryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that must be IND-CPA secure, allow certain efficient zero-knowledge proofs of knowledge and be re-blindable. Re-blindability means that there must exist a function blind, such that for every ciphertext c, $\mathsf{blind}_{\mathsf{pk}}(c; R) = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}}(c); R)$ as distributions, where R is the domain of random coins of Π. Clearly, every homomorphic cryptosystem is re-blindable, since then one can define $\mathsf{blind}_{\mathsf{pk}}(c; r) := c \cdot \mathsf{Enc}_{\mathsf{pk}}(0; r)$.

In this approach, the encrypted votes, $c_{0i} = \mathsf{Enc}_{\mathsf{pk}}(v_i; r_i)$ are posted on the bulletin board together with the zero-knowledge proof of knowledge that $\mathsf{Dec}_{\mathsf{sk}}(c_{0i})$ corresponds to a valid candidate. This zero-knowledge proof of knowledge may not be necessary, and can be replaced by a potentially simpler proof of knowledge that the voter knows $\mathsf{Dec}_{\mathsf{sk}}(c_{0i})$. At the end of the voting phase, the values c_{0i} will be mixed by ℓ, $\ell > 1$, *mix-servers* $\mathsf{MixServer}_1, \ldots, \mathsf{MixServer}_\ell$. The jth mix-server $\mathsf{MixServer}_j$ receives a list of voters encrypted votes $(c_{j-1,1}, \ldots, c_{j-1,\mathsf{voters}})$, $c_{j-1,i} = \mathsf{Enc}_{\mathsf{pk}}(v_{\chi_{j-1}(i)}; r'_{j-1,i})$, where χ_{j-1} is some permutation, and $r'_{j-1,i}$ is some random number. She then randomly re-blinds all ciphertexts and permutes them. That is, she generates a random permutation π_j, and for every $i \in [1, \mathsf{voters}]$, she creates a random blinding factor r''_{ji}. She defines

$$c_{ji} := \mathsf{blind}_{\mathsf{pk}}(c_{j-1,\pi_j^{-1}(i)}; r''_{ji}) \tag{1}$$

and writes $(c_{j1}, \ldots, c_{j\mathsf{voters}})$ on the bulletin board. This must be accompanied by a proof of correctness that for some permutation π_j and for some random r''_{ji}, (1) holds.

Every mix-server must verify the proofs of knowledge up to her round. At the end of ℓ rounds, all servers (and voters) must verify the correctness of all proofs of knowledge on the board. After that, everybody can be sure that $(c_{\ell 1}, \ldots, c_{\ell\mathsf{voters}})$ is an encryption of some permutation of $(v_1, \ldots, v_{\mathsf{voters}})$. Thus, the only thing left is to decrypt the ciphertext tuple. This can be done in a threshold manner, assuming that $\frac{1}{2}\ell + 1$ servers have to collaborate to decrypt this tuple. At the end of this section we will describe some alternative possibilities.

How to prove efficiently that (1) is true for some π_j and $\{r''_{ji}\}_i$? Next, we give a brief description of two existing verifiable shuffle protocols. (See [Nef01] for the third.)

Furukawa-Sako protocol. Represent the permutation π_j by the permutation matrix M^j, with $M^j_{ab} = 1$ iff $\pi_j(a) = b$, and $M^j_{ab} = 0$, otherwise. A nice way of using this matrix representation to achieve efficient zero-knowledge proofs is described in [FS01, Fur04]. It is based on the next fact [FS01]: Let δ_{ij} be 1 if $i = j$ and 0 otherwise. Let δ_{ijk} be 1 if $i = j = k$ and 0 otherwise. Let q be a large prime. An $\mathsf{voters} \times \mathsf{voters}$ matrix M is a permutation matrix iff

$$\sum_h M_{hi} M_{hj} = \delta_{ij} \tag{2}$$

and

$$\sum_h M_{hi} M_{hj} M_{hk} = \delta_{ijk} \ . \tag{3}$$

Thus, instead of (1), one could prove that $c_{ji} = \mathsf{blind}(\prod_{i=1}^{\mathsf{voters}} c_{j-1,i}^{M_{ji}}; r''_{ji})$ and that (2) and (3) are true.

Equation (2) can be verified by defining $s_i = \sum_{j=1}^{\mathsf{voters}} M_{ij} e_j$, for e_j chosen randomly by verifier, and then checking that $\sum_{i=1}^{\mathsf{voters}} s_i^2 = \sum_{i=1}^{\mathsf{voters}} e_i^2$. Due to (2), $s_i^2 = \sum_{j=1}^{\mathsf{voters}} M_{ij} M_{ik} e_j e_k = \sum e^2_{\chi(i)}$ and $\sum_{i=1}^{\mathsf{voters}} s_i^2 = \sum_{i=1}^{\mathsf{voters}} e^2_{\chi(i)} = \sum_{i=1}^{\mathsf{voters}} e_i^2$. Analogously, (3) is verified by checking that $(\sum M_{ij} e_j)^3 = \sum e_i^3$. Some more care has to be taken to achieve complete security [FS01, Fur04].

In this approach, the prover must make approximately $8\mathsf{voters}$ exponentiations, and the verifier must make approximately $10\mathsf{voters}$ exponentiations. When $|p| = 1024$ and $|q| = 160$, it takes about $5280\mathsf{voters}$ bits to communicate the proof of knowledge.

Groth's verifiable shuffle. An alternative, somewhat more efficient, verifiable shuffle was proposed by Groth [Gro03]. It assumes the use of an IND-CPA secure homomorphic cryptosystem Π (e.g., ElGamal, Paillier or DJ01), and of a compatible homomorphic commitment scheme. In this verifiable shuffle, the prover first commits to the shuffle. The verifier picks a vector of random integers, and the prover proves that the scalar product of this vector and the vector of encrypted votes is preserved after the shuffling. In more details, Groth's verifiable shuffle is as follows:

- Prover: For $j \in \{1, \ldots \mathsf{voters}\}$, commit to $C_{1,i} \leftarrow C_{\mathsf{pk}}(\pi(j); r_{2,j})$. Send $C_{1,i}$, together with a proof of correct shuffle, to verifier.
- Verifier: For $j \in \{1, \ldots, \mathsf{voters}\}$, generate a random t_j and send t_j to prover.
- Prover: For $j \in \{1, \ldots, \mathsf{voters}\}$: $C_{2,i} \leftarrow C_{\mathsf{pk}}(t_{\pi(j)}; r_{tj})$. Send $\{C_{1,i}\}_i$, together with a proof of correct shuffle and that this shuffle was the same as on step 1, to verifier.
- Prover proves in zero-knowledge that $\mathsf{Dec}_{\mathsf{sk}}(\prod c_{ji}^{t_{\pi(i)}}) = \mathsf{Dec}_{\mathsf{sk}}(\prod c_{j-1,i}^{t_i})$

The three first proofs of knowledge can be executed jointly, by proving that for a random λ, chosen by the verifier, $\{C_{1,i}C_{2,i}^{\lambda}\}$ commits to $\{i + \lambda t_i\}$. The proof that $\{c_i\}$ commits to $\{m_i\}$ can be done as follows: Prover sets $c_m = C_{\mathsf{pk}}(m; 0)$, for m generated by the verifier, and proves that the multiplication of the contents of $c_1 c_m^{-1}, \ldots, c_{\mathsf{voters}} c_m^{-1}$ is equal to $\prod_{i=1}^{\mathsf{voters}}(m_i - m)$. All (or at least a significant fraction) of the resulting voters zero-knowledge multiplication proofs can be done in parallel by using multi-commitments.

In this approach, the prover must perform approximately $6\mathsf{voters}$ exponentiations, and the verifier must perform approximately $6\mathsf{voters}$ exponentiations. When $|p| = 1024$ and $|q| = 160$, it takes about $1184\mathsf{voters}$ bits to communicate the proof of knowledge.

Security model and strengthening. By using a verifiable shuffle based scheme as described above, both the privacy of the voters and the correctness will hold if at least $\tau + 1$, where $\ell = 2\tau + 1$, servers are honest. It is however possible to achieve a better result. Assume that Π is the ElGamal cryptosystem and that every mix-server $\mathsf{MixServer}_j$ has additionally her own private key sk_j and public key $h_j = g^{\mathsf{sk}_j}$. Every voter encrypts his vote v as

$$(a_0, b_0) \leftarrow (g^r; v \cdot (h_1 \cdot \cdots \cdot h_\ell \cdot h)^r)$$

for $r \leftarrow_r R$. The first mix-server generates a random number r_1, and computes

$$(a_1, b_1) \leftarrow (a_0 \cdot g^{r_1}, b_0 \cdot a_0^{-\mathsf{sk}_1} \cdot (h_2 \ldots h_\ell \cdot h)^{r_1}) \ .$$

Then $(a_1, b_1) = (g^{r+r_1}, v \cdot (h_2 \ldots h_\ell \cdot h)^{r+r_1})$, that is, the first mix-server has peeled off encryption by his own key. He will then shuffle the result and accompany it with a proof of correct re-encryption and shuffling. This can be done efficiently [Fur04], although the proof will not be zero-knowledge but "permutation hiding". The second mix-server behaves analogously, by generating a random number r_2, and computing $(a_2, b_2) = (g^{r+r_1+r_2}, v(h_3 \ldots h_\ell \cdot h)^{r+r_1+r_2})$, and sending the results—in a shuffled form, accompanied with correctness proofs—to the third server. The last server outputs the set $\{(g^{r+r_1+\cdots+r_\ell}, vh^{r+r_1+\cdots+r_\ell})\}$ of encrypted votes. After that, $2t+1$ servers collaborate to recover $\{v\}$. Here, the privacy of any voter is preserved if at least one of the mix-servers is honest. At least $\tau + 1$ servers must be honest to recover $\{v\}$ from the shuffle. (See [Gro04] for a different approach.)

7 Comparison and Practical Consideration

We described shortly two main approaches to cryptographic e-voting: one, directly based on IND-CPA secure homomorphic encryption, and the second one, that is based on verifiable shuffles. (We did not describe the third major approach, based on blind signatures, due to the lack of universal verifiability. There are also potentially other problems with this approach. See, e.g., [FOO92] for one possible blind-signature based protocol.) From these two approaches, the first one is more efficient, but the second one is more universal. The verifiable shuffle-based approach becomes

more efficient when the number of candidates is large (in hundreds), when there is a need to support write-ins or different voting mechanisms (e.g., Borda). Moreover, in the verifiable shuffle-based e-voting, the voters do not have to perform zero-knowledge proofs of vote validity: it suffices to encrypt and sign the vote; invalid votes will be detected by servers anyhow. This is important in practice, since it decreases the complexity of software that needs to be installed in voters machines. Last and not least, the privacy of voters is guaranteed if at least one of the mix-servers is honest (given that the re-encryption techniques are used), while the correctness of elections is guaranteed when at least the fraction of $\frac{1}{2}$ of the servers is honest. This compares favourably to the homomorphic approach, where also the privacy depends on the threshold trust. This means, in particular, that in the case of verifiable shuffle-based solution, less servers could be used.

On the other hand, in the homomorphic e-voting protocols, the job of talliers is considerably simpler, and it is simpler to achieve universal verifiability. In the verifiable shuffle-based protocols, every mix-server has to perform Cvoters exponentiations (shuffle verification and correctness proof, re-encryption, etc), where $C \approx 20$ is a small constant. In the homomorphic protocols, the servers must just multiply the encryptions, and then jointly decrypt the result. The verification of voter's correctness proofs can be distributed among different servers so that every server verifies only a fraction of them. This means that it is likely that homomorphic protocols are faster at least by an order of the magnitude. However, one must first test this in practice. It is also likely that continuous research in both directions will result in even faster protocols. Only during the last three years, we have started to see really efficient cryptographic protocols for e-voting (e.g., protocols used in homomorphic e-voting from [DJ01, LAN02, DGS02, Lip03] and verifiable shuffle protocols from [FS01, Nef01, Gro03, Fur04]). The recent breakthroughs in both directions are at least partially caused by the recently developed efficient IND-CPA secure homomorphic cryptosystems [OU98, NS98, Pai99, DJ01, DJ03] and the relatively new concept of integer commitment schemes [FO99, DF02].

8 Further Research Topics

All described e-voting protocols have some flaws in common. Next, we outline some major problems in e-voting protocols and propose some initial solutions. An efficient solution to any of the following problems would be a major advance in the state of the art. Note that some of (or even, most of) the problems in e-voting cannot have cryptographic solutions, and we will not discuss them at all.

Information-theoretic privacy for voters. As mentioned before, ideally the privacy of voters should be information-theoretic. However, all the described approaches only guarantee computational privacy. To somewhat improve the situation, one could use public-key encryption with really high security parameter (say, ElGamal in $\mathbb{Z}_p$ with $|p| = 4096$ and $|q| = 256$). Many zero-knowledge proofs in a voting protocol can be done by using statistically hiding commitment schemes; due to statistical hiding, such proofs may executed by using moderate security parameters. Alternatively, one could try to devise protocols that really are information-theoretically secure (in a suitable trust model). At this moment the corresponding solutions are inefficient [Ots04].

Alternatively, *real* information-theoretical security can be obtained by using *cryptographic randomised response techniques (cryptographic RRTs)* [AJL04]. Here, every voter randomises his or her vote by using a publicly known probability; the result of randomisation does not say anything about the real preference of the voter. If a large number of votes are "summed" together, one can obtained an unbiased estimate to the actual voting result with a very small error margin. Cryptographic RRT of [AJL04] should be used to guarantee that the voters randomise their votes correctly. Whether this solution is politically acceptable, is unclear. However, it seems to be currently the only efficient way to guarantee unconditional vote privacy.

Eliminating the random oracle assumption. Almost all e-voting protocols use honest-verifier zero-knowledge proofs (or arguments) of knowledge that are known to be intrinsically interactive in the standard model, i.e., without any assumptions of the existence of a random oracle or a common reference string. However, for universal verifiability, the correctness proofs must be non-interactive. Honest-verifier zero-knowledge proofs of knowledge are usually made non-interactive—in the random oracle model—by using the Fiat-Shamir heuristic [FS86], by first proving that the

protocol is secure when using a random oracle, and then the random oracle with a hash function like SHA1. Unfortunately, it is known that there exist natural-looking protocols that are secure in the random oracle model but that cannot be instantiated with any function. There is no guarantee that this is not the case with the existing voting e-protocols.

The common reference string (CRS) model seems to be much more realistic, and in efficiency, protocols in the CRS model rival with the protocols in the random-oracle model. As a short example, Damgård [Dam01] has proposed the next general methodology of transforming three-round honest-verifier zero-knowledge proofs to non-interactive zero-knowledge proofs in the CRS model. Assume that all participants have an access to a trapdoor commitment public key of a central authority (e.g., the CA who is needed anyways). Then given a three-round honest-verifier zero-knowledge protocol with messages (a, e, z), the prover will first transfer a trapdoor commitment to a, obtain e, and only then return (a, z). (See [Dam01] for a complete protocol.)

In electronic voting, we however need non-interactive zero-knowledge. The current non-interactive zero-knowledge proofs in the CRS model are not that efficient, unless one wants to use non-standard assumptions. For example, Groth [Gro04] proposes efficient non-interactive zero-knowledge proofs in the CRS model, where the security assumption is that the concrete protocol is sound. It is an important open problem to design efficient non-interactive zero-knowledge proofs in the CRS model that rely only on standard computational assumptions.

Moreover, we think that the CRS model is almost realistic, but it would still be desirable to do without it. The implication of non-interactive witness-indistinguishable protocols, obtained by say derandomisation [BOV03], to the e-voting is something that must still be studied.

Achieving coercion-resistance. As noted before, an e-voting system should be secure against coercing (and vote buying). A lot of relevant cryptographic research has been focusing on receipt-freeness: that is, making it impossible for a voter to prove that he or she obeyed the coercer. However, as noted in [JJ02], receipt-freeness is insufficient. To be really coercion-resistant, an e-voting protocol should additionally be secure against the randomisation attack (coercer forces the voter to submit invalid vote), forced-abstention attack (coercer forces the voter to refrain from voting) and simulation attack (coercer buys the secret key of the voter and simulates the voter by using this key). Juels and Jakobsson proposed a *coercion-resistant e-voting protocol* [JJ02] that is secure against the mentioned attacks. However, their—yet formally unpublished—solution is not very efficient. It would be very important in practice to improve upon their protocol.

Finally, note that the next simple administrative procedure helps significantly. Allow parallel kiosk voting and Internet voting, such that for voters who have voted both ways, only their kiosk vote will be counted. However, this solution has also clear drawbacks. First, ideally, one would like to organise e-voting without any kiosk voting at all, to decrease costs. Parallel voting would instead increase the costs. Second, an invalid or a closely guarded individual is not able to go to a kiosk polling station.

Human-oriented verifiability. One huge problem with all described e-voting protocols is that they are hardly verifiable by an average Joe. To increase the trust in e-voting, it should be possible for every voter to verify that their own vote is counted correctly. There are yet no completely satisfying solutions to this problem. See [MMP02, Cha04] for some recent work in this direction, and [DJ02] for another approach that does not require trust in the equipment.

Acknowledgements.

We would like to thank anonymous referees for very useful comments. This work was partially supported by the Finnish Defence Forces Research Institute of Technology.

9 Glossary

Electronic voting: paperless voting by using any electronic or mechanical voting.

Homomorphic public-key cryptosystem: a public-key cryptosystem where group operations on ciphertexts result in group operations on plaintexts.

Internet voting: voting over Internet, by using personal computing devices.

Kiosk voting: electronic voting in predestined locations (libraries, schools).

Public-key cryptosystem: a triple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where Gen is an efficient key generation algorithm that generates a public and a secret key, Enc is an efficient encryption algorithm that uses the public key and Dec is an efficient decryption algorithm that uses the secret key.

Universal verifiability: an election is said to be universally verifiable if anybody, not only the voters, can verify that the election winner has been determined correctly.

Verifiable shuffle: a permutation of ciphertexts, such that nobody but the permuter can distinguish the used permutation, but anybody can verify that some permutation was used.

Voting mechanism: a rule to determine election winner from the votes of the voters.

References

[AJL04] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic Randomized Response Techniques. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 425–438, Singapore, March 1–4, 2004. Springer-Verlag.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO 2003, 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315, Santa Barbara, USA, 17–21 August 2003. Springer-Verlag.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Konstanz, Germany, 11–15 May 1997. Springer-Verlag.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha04] David Chaum. Secret Ballot Receipts: True Voter Verifiable Elections. *IEEE Security and Privacy*, 2(1):38–47, January—February 2004.

[CP92] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, California, USA, 16–20 August 1992. Springer-Verlag, 1993.

[Dam01] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 418–430, Innsbruck, Austria, 6–10 May 2001. Springer-Verlag.

[DF02] Ivan Damgård and Eiichiro Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. In Yuliang Zheng, editor, *Advances on Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, 1–5 December 2002. Springer-Verlag.

[DGS02] Ivan Damgård, Jens Groth, and Gorm Salomonsen. *The Theory and Implementation of an Electronic Voting System*, pages 77–99. Kluwer Academic Publishers, 2002.

[DJ01] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, 13–15 February 2001. Springer-Verlag.

[DJ02] Ivan Damgård and Mads Jurik. Client/Server Tradeoffs for Online Elections. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 125–140, Paris, France, February12–14 2002. Springer-Verlag.

[DJ03] Ivan Damgård and Mads Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In Rei Safavi-Naini, editor, *The 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364, Wollongong, Australia, July 9-11 2003. Springer-Verlag.

[El 84] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, California, USA, 19–22 August 1984. Springer-Verlag, 1985.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E82-A(1):81–92, January 1999.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secure Voting Scheme for Large Scale Elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT 1992*, volume 718 of *Lecture Notes in Computer Science*, pages 244–179, Gold Coast, Queensland, Australia, December 13–16, 1992. Springer-Verlag.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, California, USA, 11–15 August 1986. Springer-Verlag, 1987.

[FS01] Jun Furukawa and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, Santa Barbara, USA, 19–23 August 2001. Springer-Verlag.

[Fur04] Jun Furukawa. Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 319–332, Singapore, March 1–4, 2004. Springer-Verlag.

[Gib73] Allan F. Gibbard. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41:597–601, 1973.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[Gro03] Jens Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. In Yvo Desmedt, editor, *Public Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160, Miami, Florida, USA, January6–8, 2003. Springer-Verlag.

[Gro04] Jens Groth. *Honest Verifier Zero-Knowledge Arguments Applied*. PhD thesis, University of Århus, Denmark, October 2004.

[JJ02] Ari Juels and Markus Jakobsson. Coercion-Resistant Electronic Elections, 5 November 2002. Available from `http://eprint.iacr.org/2002/165/`.

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Matt Blaze, editor, *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101, Southhampton Beach, Bermuda, 11–14 March 2002. Springer-Verlag.

[Lip03] Helger Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In Chi Sung Laih, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415, Taipei, Taiwan, 30 November–4 December 2003. Springer-Verlag.

[Mat93] Yuri Matiyasevich. *Hilbert's Tenth Problem*. Foundations of Computing. MIT Press, October 1993. ISBN 0-262-13295-8.

[MMP02] Dahlia Malkhi, Ofer Margo, and Elan Pavlov. E-voting without 'cryptography'. In Matt Blaze, editor, *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, pages 1–15, Southhampton Beach, Bermuda, 11–14 March 2002. Springer-Verlag.

[Nef01] C. Andrew Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In *8th ACM Conference on Computer and Communications Security*, pages 116–125, Philadelphia, Pennsylvania, USA, 6–8 November 2001. ACM Press.

[NS98] David Naccache and Jacques Stern. A New Public Key Cryptosystem Based on Higher Residues. In *5th ACM Conference on Computer and Communications Security*, pages 59–66, San Francisco, CA, USA, 3–5 November 1998. ACM Press.

[Ots04] Akira Otsuka. An Unconditionally Secure Electronic Voting Scheme. In *DIMACS Workshop on Electronic Voting — Theory and Practice*, Piscataway, New Jersey, USA, 26–27 May 2004. No proceedings. Slides available from `http://dimacs.rutgers.edu/Workshops/Voting/`, as of June 2004.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318, Helsinki, Finland, May 31 – June 4 1998. Springer-Verlag.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, 2–6 May 1999. Springer-Verlag.

[Ped91] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, California, USA, August 11–15 1991. Springer-Verlag, 1992.

[Sat73] Mark A. Satterthwaite. *The Existence of Strategy-Proof Voting Procedures: A Topic in Social Choice Theory*. PhD thesis, University of Wisconsin, Madison, 1973.

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

[TY98] Yannis Tsiounis and Moti Yung. On the Security of ElGamal-Based Encryption. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography 1998*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, Pacifico Yokohama, Japan, 5–6 February 1998. Springer-Verlag.

[Ver04] Verified Voting Foundation. Verified Voting — Campain to Demand Verifiable Voting Results. `http://www.verifiedvoting.org`, 2004.