An Internet Voting System Supporting User Privacy

Aggelos Kiayias
University of Connecticut
aggelos@cse.uconn.edu

Michael Korman
University of Connecticut
mkorman@cse.uconn.edu

David Walluck
University of Connecticut
walluck@cse.uconn.edu

Abstract

This work introduces the ADDER *system*[1] *, an Internet-based, free and open source electronic voting system which employs strong cryptography. Our system is a fully functional e-voting platform and enjoys a number of security properties such as robustness, trust distribution, ballot privacy, auditability and verifiability. It can readily implement and carry out various voting procedures in parallel and can be used for small scale boardroom/department-wide voting as well as large-scale elections. In addition,* ADDER *employs a flexible voting scheme which allows the system to carry out procedures such as surveys or other data collection activities.* ADDER *offers a unique opportunity to study cryptographic voting protocols from a systems perspective and to explore the security and usability of electronic voting systems.*

1. Introduction

Electronic voting is currently one of the most intensely debated subjects in Information Technology. Public distrust in electronic voting has led to demands for mandatory paper audit trails, and indeed this requirement has become law in several states. In 2004, the Department of Defense canceled the Internet-based voting system SERVE that was developed by Accenture on a $22 million contract [36] because of justified security concerns raised by the academic community [24]. At the same time, the source code of currently used electronic voting systems was put under scrutiny and a multitude of flaws was unearthed [29]. The above failures threatened the credibility of e-voting and raised many questions. Should electronic voting be abandoned altogether as impossible with the current infrastructure? If not, would it be possible to build electronic voting systems that are trustworthy?

[1]The Web site of the system, which includes source code, is at: `http://cryptodrm.engr.uconn.edu/adder/`.

Although no system can claim to solve every problem facing electronic voting today, we suggest that the most obvious course of action is to develop *free and open source electronic voting systems*. While many e-voting systems exist or are under development world-wide, no fully functional and reasonably secure system is currently publicly available for free use. When the machinery used to manage an election runs inside a "black box," there is no way to verify the validity of the election.

In order to protect commercial interests, most current e-voting systems do not have publicly available source code. In some cases, code is concealed to avoid the discovery of embarrassing security flaws. Even proprietary systems that reveal source code often leave several critical components hidden. In order to ensure true democratic elections, voting software must be independently auditable and verifiable by any interested third party. For this reason, free and open source e-voting systems can be a catalyst for positive developments in the area. To the best of our knowledge, ADDER is the only free and open source e-voting system based on state-of-the-art cryptographic design.

In the remainder of the introduction, we will review the design goals, briefly summarize the operation and architecture of the ADDER system, we will provide a thorough review of the existing body of work in e-voting systems, and discuss how ADDER advances the state of the art. In section 2, we will present an overview of an ADDER election procedure. In section 3, we will discuss how our implementation captures various desired security properties. We will conclude with vulnerabilities and future directions in section 4.

1.1. Voting over the Internet

When the term *Internet voting* is used, it generally refers to remote Internet voting, where the client software communicates over the Internet to the server software, say, from a voter's PC. However, there are at least two other ways to implement voting over the Internet: kiosk voting and poll-site voting. Each of these three

ways has its own particular security requirements.

Remote In this scenario, a third party, or the voter himself (rather than election officials) has control over the voting client and operating environment.

Kiosk In this scenario, the voting client may be installed by election officials, but the voting environment is out of election officials' control.

Poll-site In this scenario, election officials have control over the voting client and the operating environment.

Although the ADDER system was designed especially for remote Internet voting, nothing prevents it from being deployed for poll-site or kiosk voting, depending on the security requirements. ADDER also has the ability to carry out small-scale and large-scale election procedures, or even surveys where strong security may be less of a concern.

It is not unreasonable to ask that remote Internet voting be as secure as voting by mail. We note that although remote Internet voting opens itself up to a wide range of attacks that may not be applicable to poll-site or kiosk Internet voting (cf. §3), it at least reduces the threat of insider attacks and allows less trust to be placed in the election officials. In many cases, voting machines arrive at polling places days or weeks early, making the threat of an on-site attack a real concern.

1.2. System overview

An ADDER election procedure is initiated through an interface which allows the administrator to provide the candidate list and specify the eligible users. Such users are *voters* and *authorities*. An ADDER election procedure progresses in the following manner. The authorities log into the system and participate in a protocol that results in the creation of a public encryption key for the system, and a unique private decryption key for each authority.

Next, each voter logs on, downloads the public key of the system, and uses that to encrypt the ballot, which is placed in an area of public storage specifically reserved for that voter. When the election is over, the server tallies the votes (using special encryption properties) and posts the encrypted result. Subsequently, the authorities provided some decoding information based on the encrypted result and their private keys. When enough such decoding information has been collected, the server combines the individual pieces to form the election result, which is then published. We note that ADDER does not employ any user-to-user communication; instead, users of the system (in particular, the authorities) communicate indirectly through the public "bulletin board" that is maintained by the system. Voters are only active in on round throughout the system's operation (unless they are also playing the role of the authorities which is possible in our architecture).

The ADDER system is implemented as a bulletin board server, an authentication server (the *gatekeeper*), and client software (either a Java applet or a stand-alone program). Figure 1 shows a diagram of the entire system.

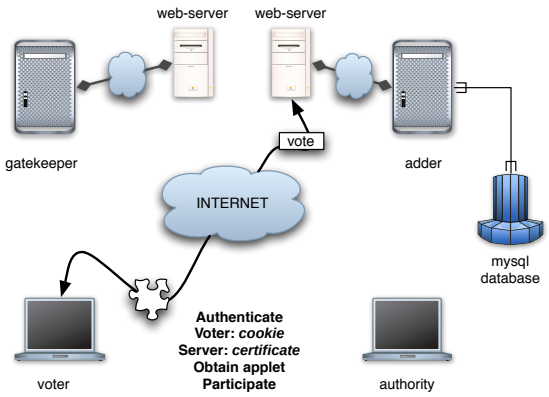

Figure 1. System diagram.

1.3. Design & security goals

In creating the ADDER system, we adhered to the following design goals.

1. *Transparency*. All of the data on the bulletin board should be accessible to the public. This includes the encrypted votes, public encryption keys, and final tallies. The bulletin board does not store secrets.

2. *Universal Verifiability*. Any election result obtained by the system should be verifiable by any third party. By inspecting the election transcript, it should be possible to perform a complete audit of any procedure.

3. *Privacy*. All voters in an election should be confident that their individual choices will remain hidden. Only the total is made available to the public.

4. *Distributed Trust*. Each procedure is "supervised" by multiple authorities, and the final sum cannot be revealed without the cooperation of a given number of authorities. Any attempt to undermine the procedure will require the corruption of a large number of authorities. Authorities and voters may overlap arbitrarily. Thus, it is possible for the voters themselves to ensure trustworthiness (or have an active role in it).

1.4. Related work

In this section, we will review existing work in e-voting systems and describe how ADDER advances the state of the art.

There exist three general design approaches for building e-voting systems based on strong cryptographic primitives: *mixnet-based*, introduced by D. Chaum [13]; *homomorphic encryption-based*, introduced by J. Benaloh [11]; and *blind-signature-based*, introduced by Fujioka et al. [19]. These approaches rely on different cryptographic primitives and, from an implementation point of view, have different advantages and disadvantages. We will briefly overview all three, discuss the known implementations that are based on them and point to their advantages and disadvantages. The ADDER system follows the homomorphic encryption-based approach.

It should be noted that some implementations do not fall into the above categories as they do not employ one of these voting-oriented strong cryptographic primitives. We will briefly discuss these schemes as well.

Mixnet-based schemes. In a mixnet-based scheme [13], the election system is built around a basic cryptographic primitive called a *mixnet*. A mixnet is comprised of a collection of servers whose task is to shuffle a given input sequence of ciphertexts (encrypted votes). This serves as an implementation of a robust anonymous channel. To ensure that mix-servers do not drop or substitute ciphertexts, it is necessary that the servers provide proofs of correct operation. A general criticism of mixnet-based schemes is that these proofs are cumbersome. We note, though, that significant advances have been made towards making robust mixnets practical [31, 20, 12, 22].

Known mixnet based implementations: (i) **VoteHere VHTi** [8, 30] is a commercial implementation which focuses on voter-verifiability. VoteHere has published their source code for public scrutiny, but it is proprietary and uses patented methods. (ii) **Scytl Pnyx** [33] is another recent commercial implementation. The source code is not publicly available, but the system appears to have been used in some government systems and small private and public sector applications in Europe. (iii) **SureVote** [14] is a commercial, special-hardware enhancement of the mixnet approach by D. Chaum which incorporates a "visual crypto" voter-verifiable component. The suggested system uses proprietary printing equipment and currently no publicly available implementation exists (we are aware that an implementation is underway [38]).

Homomorphic-based schemes. In a homomorphic encryption-based voting scheme [11], votes are added while encrypted, so no individual vote ever needs to be revealed. In order to ensure that the private decryption key of the election is not used to decrypt an individual vote, a threshold encryption scheme must be applied to distribute the key among several authorities in such a way that multiple authorities combine their shares to use it. Homomorphic encryption is the approach we followed for the ADDER system. A great advantage of this approach is that voters may openly authenticate themselves to the voting servers, i.e., there is no need for any anonymous channel to ensure voter privacy.

Two known implementations of this approach can both be found as projects based in the European Union: (i) The **CyberVote** project [2], funded by the European Commission, has developed a prototype system. The system is designed to run over the Internet, and clients can use Java-enabled mobile phones, pocket PCs, and PCs. Currently, there is no source code available for CyberVote, nor is there any downloadable software at all. (ii) **E-Vote** [21] is another EU-funded project tested in Greece in 2003. The system is based on Paillier homomorphic encryption [16]. No public implementation of the system is available and the project appears to be commercially oriented.

Blind signature-based schemes. Blind signature-based systems use a method proposed by Fujioka, Okamoto, and Ohta [19]. In this scheme, voters obtain a blind signature on their ballot from an administrator. That is, the administrator signs the ballot without being able to read its contents. Subsequently, voters submit their blindly signed ballots through an anonymous channel to a voting bulletin board that will only accept ballots signed by the administrator. The main advantage of the blind signature approach is that it removes the requirement for the anonymous channel to be robust. Its main disadvantage is that the voter needs to be active in at least two phases to ensure verifiability; it is not a "vote-and-go" voting scheme. From an implementation point of view, realizing an "anonymous channel" is not straightforward. In the known implementations listed below it is easy to correlate voters and their votes (or, in any case, there is at most a single point of failure for anonymity).

Known implementations of blind-signature based schemes: (i) **Sensus** [15] is a direct implementation of the Fujioka et al. scheme. Source code for Sensus is available, but it is released under a proprietary license. The software is no longer maintained, and the author claims that the current implementation is "clunky and won't scale" [7]. (ii) **EVOX** is another implementation based on the Fujioka et al. scheme. The first version is described by Herschberg in his Master's thesis [23], and the second version by DuRette in his Bachelor's thesis [17]. EVOX does not currently have any source code or

downloadable software available. (iii) **REVS** [25] follows the design of the second revision of EVOX. There is currently no downloadable software for REVS, but the authors have said packages will be available soon [40]. (iv) The **Votopia** project [28], created jointly by Korean and Japanese developers, was tested in the election of the MVP in the Soccer World Cup of 2002. In the Votopia system, users download a Java applet which performs cryptographic operations. A PKI is used to distribute key pairs for each server. Votopia is not publicly available and does not provide anonymity.

Other methods. There exist other implementations not based on voting-oriented cryptographic primitives. We now review them briefly.

The **Diebold AccuVote-TS** system is one of the most heavily criticized non-Internet-based electronic voting systems used in practice [29]. Problems pointed out include incorrect use of cryptography, poor code quality, and possibility of smartcard forgery, among many others. Despite Diebold's rebuttal [9], the system is mistrusted by a number of experts.

The **SERVE** system is a Department of Defense government-funded project for Internet-based voting. SERVE works as follows. For each voting district, a local election official (LEO) generates a key pair. When a ballot is cast, it is sent with identification over the Web. This information is encrypted with the Web server's public key. The Web server verifies the eligibility of the voter, decrypts the ballot, removes the voter's name, and re-encrypts the ballot with the LEO's public key. This ballot is then sent to the LEO. SERVE was found to have many vulnerabilities [24] and the project was discontinued [36]. One of the major vulnerabilities particular to SERVE is that the Web server knows the vote of each voter, and can tie it to his identity. If the Web server is compromised, voter privacy is broken entirely.

RIES (Rijnland Internet Election System) [6] is an election system developed in 2003 and 2004 for the Water Board elections at Rijnland and De Dommel in the Netherlands. The system has many vulnerabilities [5] such as the use of a single master triple-DES key.

EVM2003 [3] is a project to develop a free and open source electronic voting machine. However, it seems to have undergone very little activity since its inception in 2003 and does not seem to employ any cryptographic voting protocols.

Condorcet Internet Voting Service (CIVS) [1] is a Web-based free voting system that employs the Condorcet election method. Voters submit a ranking of candidates instead of picking only one candidate. CIVS employs some cryptographic integrity mechanisms but falls short of offering cryptographic guarrantees for voter privacy.

GNU.FREE [4] is a free Internet voting system released by the GNU project. In GNU.FREE, voting is not done over the Web. Rather, a stand-alone Java program is used to cast votes which are encrypted using a cipher (BlowFish). The system does not provide sufficient security (beyond preventing regular eavesdropping), and it is easy for a malicious system to correlate voters and their votes. It is worth noting that EVM2003, CIVS, and GNU.FREE are the only voting systems we have found that are free software.

1.5. Comparison to the present work

Adder is a Internet-based e-voting system based on a strong voting-oriented cryptographic primitive (homomorphic encryption). Adder is free software released under the GNU GPL. Anyone can create his own installation of Adder for testing or general usage. To the best of our knowledge, Adder is the first system of this kind.

Moreover, Adder compares particularly favorably against commercial Internet voting systems (e.g., SERVE). For instance, Adder supports large-scale trust distribution for voter privacy. As a large number of key-share-holding authorities is supported, elections can essentially be run by the community. In addition, Adder employs state-of-the-art encryption methods and puts forth the very attractive design principle of transparency: the bulletin board is publicly readable and holds no secrets (thus even if compromised the privacy of the voters cannot be violated). Additionally, the whole election process is universally verifiable. Admittedly, Adder has many limitations; nevertheless these are shared by all systems of the same kind (cf. §4).

While there exist serious and justified security concerns regarding the employment of Internet-based voting for sensitive election procedures such as Presidential elections, we believe the existence of free and open source system like Adder will motivate further testing and development, and will be a step forward towards the development of truly robust and trustworthy e-voting procedures.

2. Overview of an election

At the heart of the system operation is a "bulletin board" (a notion introduced in e-voting by Benaloh [11]). The bulletin board is a "public channel" with memory, where all authenticated users (voters and authorities) can append data. We realize the bulletin board as an SQL database which provides availablity and data integrity, while authentication is performed by a Kerberos-like "gatekeeper" server, which assigns cryptographic tokens to users that allow the users to perform

various actions upon receiving proper credentials. Client software, either a Web browser with a digitally-signed Java applet, or a stand-alone GUI program, is used to connect to the system.

There are three types of users in the Adder system: *authorities*, *voters*, and *administrators*. Authorities are responsible for jointly maintaining the security and privacy of the election, voters are the users who actually cast ballots, and administrators are responsible for creating and managing elections.

A typical Adder election procedure consists of several stages, outlined below. See Figure 2 for a graphical depiction of the interactions between parties.

Procedure creation. An election begins when an administrator logs onto the bulletin board server and submits the procedure creation data. Among the parameters specified are the procedure identifier, the identities of voters and authorities that are eligible to participate, the authority threshold t, the minimum and maximum number of candidates that voters can select, the list of candidates, and the election duration. Once the administrator submits this form, the bulletin board server populates the database with the corresponding information.

The bulletin board server subsequently generates the values p, q, g, and f where p is a "safe prime" i.e., $p = 2q + 1$ and q is a prime, as well and g is an order q element of $\mathbb{Z}_p^*$ as well as $f \in \langle g \rangle$. Below we denote by $\mathcal{E}_h : \mathbb{Z}_q \times \mathbb{Z}_q \to \langle g \rangle^2$ the mapping $(r, x) \to \langle g^r, h^r f^x \rangle$. (this is a form of a ElGamal encryption [18]). All of these parameters (and all cryptographic data generated by the server in the future) are stored on the bulletin board. Henceforth, the values p, q, g, and f are referred to simply as the "cryptographic parameters." They can be used for many election procedures.

In the following three steps the set of authorities that are enabled for the procedure will jointly produce the public key of the system initializing a threshold encryption scheme (cf. [32]). We note that not all authorities are required to successfully carry out the steps. In the steps below, we will use the notation $Auths_{stage1}$, $Auths_{stage2}$, and $Auths_{stage3}$ to denote the subsets of authorities that succeed in completing the stages of the system public key generation. If the authorities that complete all three stages are below a safety threshold t', the system terminates the procedure. Note that the safety threshold satisfies $t' > t$, where t is the distributing trust parameter.

Authority public key generation. Once the procedure has been created, the authorities collaborate to create the public encryption key of the system. At the end of the election, they each contribute a part to the decryption of the result. Note that no authority has the ability to decrypt a single vote because the actual private key of the system does not exist in the private memory of any one entity. Rather, it is broken up in the form of the authorities' individual private keys. In order to decrypt a single vote, an amount of authorities greater than the authority threshold t would all have to collaborate. To ensure that no malicious coalition can exceed such threshold in a certain deployment, election officials can designate authorities with differing political interests, so they would have no incentive to collude.

An authority i downloads the cryptographic parameters from the bulletin board. It then generates an ElGamal key-pair $\langle h, x \rangle$, where $h_i = g^{x_i}$, and $x_i \in_R \mathbb{Z}_q$. In this step, h_i serves as the public key of the authority, and x_i serves as the private key. The public key is stored on the bulletin board. Once all authorities have completed this stage, the public key of each authority is stored on the bulletin board. Let $Auths_{stage1}$ be the set of authorities that have completed this step. If $|Auths_{stage1}| < t'$ (the safety threshold), then the server will terminate the procedure here.

Polynomial generation. At this stage, the authorities will participate in a protocol that will compute the public key for the election using a distributed key generation protocol.

An authority i logs in, reads h_j (the public keys) for all authorities $j \in Auths_{stage1}$, and the authority threshold t. The authority creates a polynomial $P_i(x) \in \mathbb{Z}_q[x]$, and publishes the value $\mathcal{E}_{h_j}(r_j, P_i(j))$ for each $j \in Auths_{stage1}$, where $r_j \in_R \mathbb{Z}_q$. This serves as a form of encryption of authority j's index evaluated in authority i's polynomial. Finally, he publishes $G_{i,\ell}$ for $\ell = 0, \ldots, t-1$. Let $Auths_{stage2} \subseteq Auths_{stage1}$ be the set of authorities that have completed this step. If $|Auths_{stage2}| < t'$, the server terminates the procedure.

Private key generation. Each authority $j \in Auths_{stage2}$ connects to the bulletin board server and reads $\mathcal{E}_{h_j}(r_j, P_i(j))$ for all $i \in Auths_{stage2}$. Note that authority j can decrypt all of these values, as they were encrypted with his public key. Thus, he retrieves x_j (his private key) from his private memory. He decrypts these values, computes their sum, and stores the result in private memory. At this point, authority j has the sum of the evaluations of his index in every other authority's polynomial. Let $Auths_{stage3} \subseteq Auths_{stage2}$ be the set of authorities that have completed this step. If $|Auths_{stage3}| < t'$, the server terminates the procedure.

Public key publication. The bulletin board server publishes the value $h = \prod_{i \in Auths_{stage2}} G_{i,0}$. Note that h is the sum of all of the authorities' polynomials evaluated at 0. This value h serves as the public key of the procedure, which voters will use when they encrypt their ballots.

Voting. Now, the election may begin. A voting timer

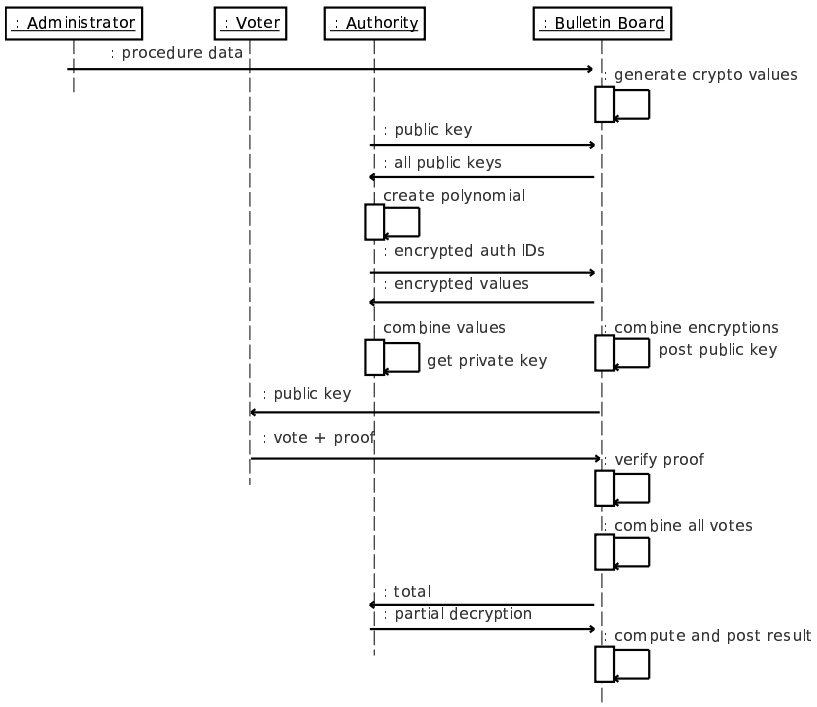

Figure 2. The stages of an Adder election procedure. Note that the Gatekeeper is involved in every transaction that requires user authentication.

is set which will expire when the election duration is reached. Each voter i downloads the public key h from the bulletin board, as well as the cryptographic parameters. The voter is permitted to vote between K_{min} and K_{max} candidates, where $0 \leq K_{min} \leq K_{max} \leq L$, and L is the total number of candidates. The voter thus forms a ciphertext for each candidate, encrypting 1 if the voter votes for that candidate, and 0 otherwise. The encrypted vote is therefore a vector of ciphertexts, one for each candidate, each one of the format $\langle g^r, h^r f^\zeta \rangle$ where $\zeta\{0, 1\}$. Observe that the encryption satisfies the homomorphic property: $\langle g^{r_1}, h^{r_1} f^{\zeta_1} \rangle \oplus \langle g^{r_2}, h^{r_2} f^{\zeta_2} \rangle = \langle g^{r_1+r_2}, h^{r_1+r_2} f^{\zeta_1+\zeta_2} \rangle$. Along with the encrypted vote, the voter computes a proof of ballot validity P which is a non-interactive zero-knowledge proof that the encrypted vote is properly formed. In this case, "well-formed" means that the voter has voted for the correct number of candidates, and that each ciphertext is the encryption of either 0 or a 1. Finally, the pair $\langle V, P \rangle$ is posted on the bulletin board.

The bulletin board server then verifies the proof. If the proof is valid, and the voter has not already voted, it posts the vote and proof on the bulletin board. Otherwise, it discards the vote and proof and returns an error.

Result tabulation. When duration of the election has expired, or when an administrator manually ends the election, result tabulation occurs. The bulletin board contains received a vote V_i from each voter i. The encrypted sum of the votes is computed by the bulletin board server combining the votes as

$$V = \bigoplus_{i=0}^{m} V_i,$$

where $\oplus$ is the homomorphic operation on votes, and V is a vector of ciphertexts encrypting the total number of votes for each candidate. This combined sum is then posted on the bulletin board.

Authority decryption. Now, each authority logs in again, downloads the encrypted result of the election, and submits his partial decryption. These partial decryptions are posted on the bulletin board.

Result decryption. Once each authority has submitted his partial decryption, the bulletin board server combines the partial decryptions and decrypts the result.

Result publication. The end result is finally published on the bulletin board. The election is over, and no more logins are accepted. Anyone can now view the final result.

3. Electronic voting concerns

After the major controversies with electronic voting, several scientists came out rather harshly against electronic voting, and this is understandable given the flaws in current voting systems. More recently, however, researchers have begun to take a more serious look at cryptographic voting systems and their feasibility, even going so far as to express optimism about electronic voting [27]. In this section, we describe how ADDER address common concerns in electronic voting.

3.1. User authentication

ADDER performs user authentication by employing a Kerberos-like system called the *gatekeeper*. Actions that can be performed by users include: creating procedures, deleting procedures, voting, authority actions, and resetting procedures. The gatekeeper keeps a private table of user credentials, as well as a private signing key. When a user wishes to perform an action in the ADDER system, he authenticates himself (either by password or public-key authentication where it makes sense and already exists (e.g., the military) to the gatekeeper. The gatekeeper then ensures that the user is eligible to perform the requested action, and if so, generates a signed *ticket* that authorizes the user. The ticket is then presented to the bulletin board server, which verifies the gatekeeper's signature, and allows the user to perform the requested action. In the implementation this transfer from the gatekeeper server to the bulletin board server is essentially transparent from the user's point of view.

As a result of this authentication system, ADDER achieves a realization of the bulletin board as an entity that holds no secrets, but solely enforces constraints on the posting of messages and ensures the integrity of the election audit trail. Secret data is separated into the gatekeeper component of the system. The bulletin board, which holds all data relevant to the election, is thus in complete public view. Furthermore, it is possible to have several gatekeepers, each responsible for authenticating one segment of users, or as a measure of redundancy in case of failure. At the same time the database of the bulletin board server can also be replicated.

To enter users into the system, the administrator provides the gatekeeper with a list of names and email addresses. The system can then randomly generate passwords and email the users their respective credentials. Additionally, printed letters can be produced, if administrators would like to notify users by mail. If voter anonymity is desired, voter identifiers can be assigned randomly by the gatekeeper and the system can use those instead of any identifying information (pseudonymity). The identity of the voter can be authenticated, but cannot be known without collusion between the bulletin board server and the gatekeeper. Since random user identifiers may be assigned independently of the bulletin board server, so that even if the votes are one day compromised, the identity of the user cannot be learned or proved directly from the bulletin board data.

3.2. Ballot privacy

In most elections, it is crucial that the privacy of each voter is maintained. It should not be possible for anyone to determine how each voter voted. This is in conflict with the goal of universal verifiability, since determining that each voter voted correctly and tallying the votes both require access to the votes themselves. To address this problem, ADDER goes around these problems by employing homomorphic encryption techniques. Recall that an encryption function $\mathcal{E}$ is homomorphic if, given $\mathcal{E}(x_1), \mathcal{E}(x_2)$, where x_1 and x_2 are plaintexts, it is possible to compute $\mathcal{E}(x_1 + x_2)$. That is, there is an operation $\oplus$ such that $\mathcal{E}(x_1) \oplus \mathcal{E}(x_2) = \mathcal{E}(x_1 + x_2)$. Given a homomorphic encryption function, it is possible to add a sequence of encrypted votes without being able to read the votes themselves. Thus, once all of the votes are cast, any independent third party can perform the summation themselves.

3.3. Universal verifiability

The ADDER system is designed to be "universally verifiable." All data present on the bulletin board are publicly viewable. The bulletin board server runs with a special read-only account designated for auditors. The free and open source verification suite that accompanies the system performs the following tasks:

1. *Tallying of the encrypted ballots*. Since homomorphic encryption is used, there is no need to possess a private key to add the votes. All that is required are the encrypted ballots themselves. The verification suite repeats the server-executed encrypted ballot aggregation.
2. *Verification of all proofs*. Proofs of ballot validity are checked for each voter.
3. *Decryption of the final tally*. Once all of the authorities have finished submitting their partial decryptions, the verification suite recomputes the Lagrange coefficients and decrypts the final sum.
4. *Verification of the hash chain*. In order to enforce the integrity of the bulletin board and causality of events, we employ a hash chain. A hash of the database is computed at the start of an election, and

stored in the database. At frequent intervals (say, every five minutes), a new hash is computed, incorporating the previous hash. If anyone is to tamper with the database, it must be done between hash snapshots, since each successive hash guarantees the integrity of every snapshot that has occurred before it. The hash chain is verifiable since each ballot in the database is accompanied by a timestamp. The entire history of the database can then be reconstructed, and the hash chain can be computed and compared with the published value.

3.4. Voter verifiability

Voter verifiability is a common concern in electronic voting systems. Many *direct recording electronic* (DRE) systems use a *voter-verifiable paper audit trail* (VVPAT). Before the vote is counted in the machine, a paper confirmation containing the voter's ballot is printed and displayed to the voter. If the voter agrees with the print-out, he indicates this to the machine, and the paper ballot is dropped into a tamper-proof box. If the voter disagrees with the print-out, the paper ballot is discarded, and the voter can re-vote. The goal of a VVPAT is to ensure that there exists a paper record of every vote that is counted by the electronic machines. Furthermore, each item in the paper trail has been certified by the voter who cast it. If the electronic tally is disputed, it is thus possible to perform a manual recount of the votes by inspecting the paper trail.

The focus of the ADDER system is on providing a strong cryptographic solution to electronic voting. As such, we feel that voter verifiability remains an orthogonal concern. If ADDER is used in a controlled setting, at a designated polling place, it is an easy matter to add a VVPAT component. However, it must be noted that VVPAT makes no sense in the context of remote Internet voting. Since voting is performed on the voter's computer, it is not possible for an authentic paper trail to be maintained. It thus remains the decision of the organization conducting the election whether or not a VVPAT is a necessary component, as ADDER provides the flexibility to accommodate both large and small-scale elections.

4. Vulnerabilities and future directions

ADDER, as an Internet-based voting system, is susceptible to a number of vulnerabilities and attacks. We list them below, along with possible solutions which will be implemented in future versions of the ADDER system.

Distributed key generation. Currently, the distributed key generation subsystem of ADDER is not universally verifiable, i.e., some authorities may misbehave without being detected. This may result in a bias in the randomness of the elections' public keys. There are standard cryptographic solutions to this problem which will be incorporated into future versions.

Transcript availability and robustness. The ADDER system relies on an SQL database for the preservation of the election trail which includes all encrypted ballots and public keys of authorities, as well as other auxiliary values. While this database contains no secrets, (due to our transparency design principle) and thus we need not provide any access control for read access, it is supposed to be available for the verification of the election, as well as the publication of the election results. An insider attack that deletes the database in some stage of ADDER's operation would naturally be destructive to an election process. To tackle this problem, the database can be replicated and stored at frequent intervals in remote replication database servers. Future versions of ADDER will incorporate such fault-tolerance techniques.

Vote buying and coercion. In the current implementation, a malicious voting client may produce a proof of how a user voted or otherwise leak information about the voter. Future versions of ADDER will incorporate the use of ciphertext re-randomization, a technique that has been suggested as a way to obtain protection against such proof-of-vote attacks in the context of e-voting (cf. [10]).

Voter verifiability. Currently, our system does not offer to a voter method for physically verifying that his published encrypted ballot encrypts his actual choice. Instead, the voter relies on the correctness of the client software for this task. We note that dealing with this is a complex problem, since any method for voter-based verifiability can also potentially used by the voter to prove how he voted and thus allow for vote buying.

Viruses and other client-environment hazards. Although cryptography cannot entirely solve the secure platform problem for remote Internet voting [34], an obvious area where it would help is by employing a trusted computing environment [35]. It is typical for many cryptographic protocols to assume an ideal environment that is free of viruses and other malicious software and frequently many assumptions are unspecified [26]. In other cases, where operating details are specified, they are often too-optimistic or ignore issues such as security and usability tradeoffs.

If ADDER is run on a standard PC, the client environment can be subverted by viruses and other malicious code which may reside on it and have taken control of the local machine. Such attacks are particularly devastating for any e-voting scheme employing a non-dedicated PC environment. A standard approach to deal with such attacks is to provide a stripped down operat-

ing environment containing an authorized Web browser with the ADDER applet installed which can be downloaded and burned into a bootable CD-ROM. With such a CD, the user can boot into a clean environment. Using a stripped-down version of a free and open source operating as a base would allow complete auditing of the software down to the compiler and operating system level, which is important as the electronic voting system can only be as trustworthy as its compiler [37]. In a future version of ADDER, we will consider the development of such a stripped down operating environment.

Denial-of-service attacks. The ADDER system, as all Internet-based voting systems, is susceptible to denial-of-service attacks. While such attacks can be particularly devastating against a voting system, there exist countermeasures which we will consider incorporating into future versions of ADDER (a possible approach to consider is [39]).

User interaction. Computer systems running election protocols may in one way or another engage their human operators to become active participants in security/integrity verification (cf. [14]) and the system's security guarantees may rely partly on users reporting certain types of abnormal operation. Enabling human verifiability of secure system operation is an important goal for any secure system implementation.

References

[1] Condorcet Internet Voting Service. *http://www.cs.cornell.edu/andru/civs.html*.

[2] CyberVote. *http://www.eucybervote.org/*.

[3] Evm2003. *http://evm2003.sourceforge.net/*.

[4] GNU.FREE: Heavy-Duty Internet Voting. *http://www.j-dom.org/users/re.html*.

[5] RIES and Better. *http://www.surfnet.nl/bijeenkomsten/ries/salomonson.ppt*.

[6] RIES facts and features sheet. *http://www.surfnet.nl/bijeenkomsten/ries/RIES_Word1.doc*.

[7] Sensus. *http://lorrie.cranor.org/voting/sensus/*.

[8] VoteHere VHTi: Frequently Asked Questions. *http://votehere.net/vhti/documentation/*.

[9] Diebold Election Systems: Checks and Balances in Elections Equipment and Procedures Prevent Alleged Fraud Scenarios. Technical report, July 2003.

[10] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Principles of Distributed Computing*, pages 274–283, 2001.

[11] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1987.

[12] D. Boneh and P. Golle. Almost Entirely Correct Mixing with Applications to Voting. In V. Atlury, editor, *Proceedings of the 9th ACM Conference on Computer and Communication Security (CCS-02)*, pages 68–77, New York, Nov. 18–22 2002. ACM Press.

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

[14] D. Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[15] L. Cranor and R. Cytron. Design and Implementation of a Practical Security-Conscious Electronic Polling System. Technical Report WUCS-96-02, Washington University, January 1996.

[16] I. Damgård, J. Groth, and G. Salomonsen. *Secure Electronic Voting*, chapter 6, pages 77–99. Kluwer Academic Publishers, 2003.

[17] B. W. DuRette. Multiple Administrators for Electronic Voting. Bachelor's thesis, MIT, 1999.

[18] T. Elgamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[19] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Proceedings of AUSCRYPT '92*, pages 244–251, 1993.

[20] J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle. In J. Kilian, editor, *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.

[21] J. Gilberg. E-VOTE: An Internet-based Electronic Voting System: Consolidated Prototype 2 Documentation. Technical Report e-VOTE/WP-7/D7.4/3.0/29-05-2003, May 2003. *http://www.instore.gr/evote/evote_end/htm/3public/doc3/public/public_deliverables/d7_4/Consolidated_Docu_final.zip*.

[22] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels. Optimistic Mixing for Exit-Polls. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, volume 2501, pages 451–465, 2002.

[23] M. A. Herschberg. Secure Electronic Voting Over the World Wide Web. Master's thesis, MIT, 1997.

[24] D. Jefferson, A. Rubin, B. Simmons, and D. Wagner. A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE). Technical report, 2004. *http://servesecurityreport.org/*.

[25] R. Joaquim, A. Zúquete, and P. Ferreira. REVS – A Robust Electronic Voting System. *IADIS International Journal of WWW/Internet*, 1(2), December 2003.

[26] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *Proceedings of the 14th USENIX Security Symposium*, pages 33–50.

[27] C. Karlof, N. Sastry, and D. Wagner. The promise of cryptographic voting protocols. *http://www.cs.berkeley.edu/ daw/papers/cvop-unpub05.pdf*.

[28] K. Kim. Killer Application of PKI to Internet Voting. In *IWAP 2002*. Springer Verlag, 2002. Lecture Notes in Computer Science No. 1233.

[29] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach. Analysis of an Electronic Voting System. In *IEEE Symposium on Security and Privacy*, May 2004.

[30] C. A. Neff. Verifiable Mixing (Shuffling) of ElGamal Pairs.

[31] C. A. Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 116–125, Philadelphia, PA, USA, Nov. 2001. ACM Press.

[32] T. P. Pedersen. A Threshold Cryptosystem without a Trusted Party (Extended Abstract). In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer-Verlag, 8–11 Apr. 1991.

[33] A. Riera and P. Brown. Bringing Confidence to Electronic Voting. *EJEG*, 2(1), July 2004.

[34] R. L. Rivest. Electronic voting. In *Financial Cryptography '01*, volume 2339 of *LNCS*, pages 243–268. Springer-Verlag, 2001.

[35] A. D. Rubin. Security considerations for remote electronic voting. In *29th Research Conference on Communication, Information and Internet Policy (TPRC2001)*, October 2001.

[36] J. Schwartz. Online Voting Canceled for Americans Overseas . The New York Times, February, 6 2004.

[37] K. Thompson. Reflections on trusting trust. *Commun. ACM*, 27(8):761–763, 1984.

[38] P. Vora. Citizen Verified Voting: An implementation of Chaum's voter verifiable scheme. Talk given at the DIMACS Workshop on Electronic Voting, Rutgers U., NJ, May 26-27, 2004.

[39] X. Wang and M. K. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions. In *2003 IEEE Symposium on Security and Privacy*, pages 78–92.

[40] A. Zúquete. Private communication.