Every Vote Counts: Ensuring Integrity in Large-Scale DRE-based Electronic Voting

Feng Hao
School of Computing Science
Newcastle University, UK
feng.hao@ncl.ac.uk

Matthew Nicolas Kreeger
Thales Information Technology Security
Cambridge, UK
matthew.kreeger@thales-esecurity.com

***Abstract*—This paper presents a new and complete cryptographic e-voting system, called Direct Recording Electronic with Integrity (DRE-i). The DRE is a widely deployed voting system that commonly uses touch-screen technology to directly record votes. However, a lack of tallying integrity has been considered the most contentious problem with the DRE system. In this work, we take a broad interpretation of the DRE: which includes not only touch-screen machines, as deployed at polling stations, but also remote voting systems conducted over the Internet or mobile phones. In all cases, the system records votes directly. The DRE-i protocol is generic for both on-site and remote voting; it provides a drop-in mathematical solution to ensure tallying integrity even if the DRE machine is completely corrupted. Besides the tallying integrity, we also describe procedural means to protect voter's privacy in a complete system. As compared with the currently well-known Helios e-voting system, our work represents a significant improvement in two main aspects. First, it permits a thin client: a web-based implementation of DRE-i does not require any Java plug-in to be installed or Javascript to be enabled. Second, it is self-tallying: as we adopt a novel technique to encrypt votes, anyone can tally votes by simply multiplying ciphertexts without needing any private keys or tallying authority involvement.**

I. Introduction

The Direct Recording Electronic (DRE) voting system commonly adopts touch-screen technology to record the voter's choice directly. The system can provide several benefits in terms of usability, accessibility and efficiency [10]. Voters, including the disabled and the elderly, generally consider a touch screen interface easy-to-use [8] and the electronic display can be conveniently customized to various language options. In addition, DREs can effectively limit voters to select only a specified number of candidates, hence preventing both over- and under-voting [10].

The procedural complexity of a DRE is low. Firstly, the voter authenticates himself at the polling station and obtains a token (typically, a PIN slip or smart card) [2]. The voter enters a private booth, presents the token to the DRE machine, and starts the voting process. Figure 1 shows an example[1] of the selection choices on the touch screen. The voter follows two basic steps to cast a vote: 1) select a candidate; 2) confirm or cancel. If the voter opts to "confirm" the intended vote, the vote is recorded. Otherwise, no vote is recorded, and the screen will again prompt the voter to select the desired candidate.

The perceived benefits of DRE had created a wave of adoption in many countries. For example, the use of DRE technology has expanded rapidly in the United States since the 2000 election – from 12% in that election to 29% in 2004, to 38% in 2006 (but it quickly dropped to 32% in 2008 for reasons we will explain) [36]. This was largely attributed to the Help America Vote Act (HAVA) of 2002, which requires at least one voting machine in each precinct to fully accommodate disabled voters, and DREs are arguably the only system that meets this requirement. DRE technology is also widely used outside the USA. For example, India moved to fully DRE voting in the 2004 election and Brazil started its first fully DRE election in 2002 [35].

However, potential security vulnerabilities with DREs were publicized as a result of several studies. The analysis of the Diebold voting system carried out by Kohno et. al. was one of the first, and highly influential [2]. The researchers had access to the source code of the system, which was available on the Internet. They discovered serious system flaws and software vulnerabilities. Other studies revealed similar results [11]. The alarming level of security defects found casted wide-spread doubt on the integrity of the tallying result. In response to the research findings in [2], [11], many people called on the government to abandon DREs completely and to discard e-voting in general. Several states in the US consequently reverted to old-fashioned voting machines, resulting in the quick decline of DRE usage from 38% in 2006 to 32% in 2008 [36].

E-voting is a new technology, and as with any new technology, it takes time for it to develop and improve. Rushing to embrace e-voting is just as harmful as rushing to reject it. In this study, we first need to distinguish protocol errors from implementation errors. Many of the reported problems, such as buffer overflow, SQL injection etc, are related to the latter. However, from a system point of view, we should be more concerned with the former, because protocol errors are more fundamental and harder to fix.

In the proceeding example (see Figure 1), it contains a fatal protocol error (although similar systems have been widely deployed). After the voter casts a vote, there is no way for the voter, or others, to verify whether the vote has been correctly counted in the final tally. The voter has to

[1]This is the simplest example and for illustration only. The real system can be more complex, but the basic procedure is roughly the same.

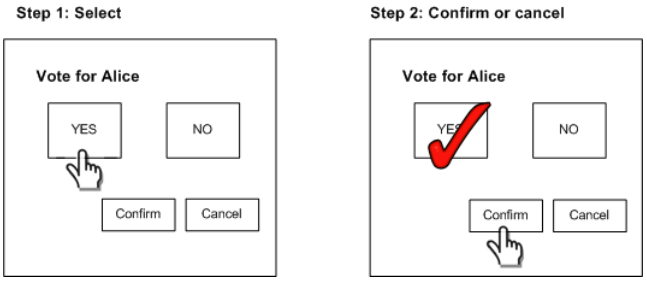

Figure 1. A touch screen based single-candidate DRE voting system

completely trust the DRE machine. This is unacceptable from a security point of view for the simple reason that a totally trustworthy third party does not, and will not, exist. A practical countermeasure adopted by many countries is through government certification. But this fails to resolve the protocol error. Whilst the certification may be necessary, it is not sufficient in building up public confidence and trust. Previous studies have shown that even certified machines still contain an abundance of software defects and system vulnerabilities [2], [11].

In the paper, we attempt to provide a drop-in mathematical solution to ensure tallying integrity in a DRE system, without altering the voter's intuitive voting experience. While the solution primarily focuses on tallying integrity, we also present procedural means to protect the voter's privacy in a complete system. Our objective is not to propose an "ultimate solution" for use in all elections. But rather, we aim to explore, through cryptographic means, how far we can achieve in electronic voting in terms of security, performance and simplicity. We believe it is crucial to understand the exact merits and limits of e-voting before deciding where it is applicable.

II. Past Work

In this section, we review past work related to electronic voting. There are many paper-based voting protocols, such as Prêt à Voter [7], Scantegrity [6], ThreeBallot [5] etc. These protocols have different security properties and trade-offs when compared with e-voting systems. It is, however, beyond the scope of this paper to evaluate these protocols.

There are two categories of e-voting: decentralized and centralized [12]. In the former case, the election is run by voters themselves without involving any trusted third parties. A decentralized e-voting protocol can provide the theoretical best protection of the voter's privacy: the voter does not have to trust anyone but himself [12]. However, such protocols are limited in terms of scalability and are only suitable for small-scale elections [12], [19], [20]. This contrasts with centralized e-voting, where centralized administration ensures a greater level of robustness and is considered more suitable for large-scale elections [8], [24].

DRE is one example of centralized e-voting. As with any centralized system, the DRE machine becomes an attractive target of attack and a single point of failure. If the machine crashes in the middle of election, it may cause great disruptions to the election. On the other hand, due to the centralized nature, it is considerably easier to focus resource on protecting the machine from system failures.

Apart from system robustness, a key issue with a DRE system is whether the software is trustworthy. Government certification of the DRE is one perceived method to instill trust [2], however, numerous studies have shown that it is imprudent to rely on certification for establishing trust [2], [11].

To build a trustworthy DRE system, there are two general approaches: through trusted computing [3], [4], [10] or mathematics [8], [18]. The first approach attempts to bootstrap trust from the integrity of a small piece of hardware and software, called Trusted Computing Base (TCB) [3]. However, the existence of the TCB is sometimes called into question [24]. For example, Scytl is a commercial TCB-based solution [30]. It is an external device that can be attached to the DRE machine, permitting voters to verify votes in real-time. Essentially, this solution shifts trusting the DRE to trusting Scytl – voters must completely trust the software of the Scytl device (and trust it does not collude with the DRE) [24].

The second approach is more promising. Rivest once suggested a famous design principle for e-voting systems: that is "software independence" [31]. This principle states that it does not really matter how the software is written inside the system, by verifying the output of the software, the voter can get assurance that the software is tallying votes correctly. Essentially, this approach shifts from trusting software to trusting mathematics – and mathematics is publicly verifiable. The "software independence" is also the guiding principle in our work.

There have been many cryptographic voting protocols proposed in the past, for example [8], [25], [26], [32], [33]. In general, they all involve tallying authorities (also called trustees) and employ mix-nets or threshold decryption. For example, in the VoteBox voting system, all the votes are encrypted by a tallying public key using ElGamal encryption [8]. The private key is shared among a number of authorities using a secret sharing scheme. The homomorphic property of the ElGamal encryption facilitates adding votes in the ciphertext form. The final tally is revealed when a quorum of authorities are reached to reconstruct the decryption key. In a different approach, Chaum proposes to let each trustee possess his own public key, which in turn is used to encrypt the votes [25]. The encrypted votes are run through a series of mix-nets for shuffling and re-encryption before they are finally decrypted. Neff has a similar mix-net based protocol [26].

The latest development in this line of research is the web

implementation of the Helios e-voting system [32], [33]. The Helios system is built on pre-existing cryptographic techniques and web development tools. The system design is similar to past works [8], [25], [26] – it depends on tallying authorities, and employs mix-nets in Version 1.0 [32] and threshold decryption in Version 2.0 [33]. To some extent, the Helios system reflects the state of the latest cryptographic voting research.

The primary goal of the Helios design is to provide cryptographic assurance on tallying integrity, as stated in the Helios paper [32]: "Trust no one for integrity, trust Helios for privacy." Although Helios adopts Java to encrypt the vote in a client browser, as demonstrated in [39], the encryption provides no assurance on vote secrecy, since the browser may be compromised.

The practical significance of the Helios system is that it has been used in some real-world elections. In 2009, the Université catholique de Louvain (UCL) adopted a customized version of Helios (v2.0) to elect its president [33]. Subsequently, the International Association for Cryptologic Research (IACR) chose Helios 2.0 to run a trial election in 2010 [34]. Both elections were reported a success [33], [34].

The elections demonstrated two advantages of e-voting: precision and convenience. It is interesting to note that in the UCL election, the leader came short of winning the election in the first round by only 2 votes out of a total of about 4000 [33]. The audit of the tally led to a quick acceptance without any recount or dispute. This level of precision is remarkable when compared to the often tedious and error-prone manual counting. In another election, the IACR members were asked whether to switch to electronic voting or keep the current paper-based system. The vast majority of members voted for switching to electronic voting (344 vs 32) [34]. Many members found the e-voting system convenient to use, when compared to the IACR's traditional voting system, based on double envelopes sent via postal mail.

However, practical deployments have also revealed some (inherent) drawbacks of the Helios system. These drawbacks are also generally applicable to other cryptographic voting protocols. Although Helios is customized to be web-based, the lessons are relevant to touch-screen based implementations too.

First, the use of a Java plug-in was commonly seen as a major drawback. Helios requires a trustworthy client execution environment. In particular, it requires the user to install a Java plug-in in the browser (Helios uses LiveConnect to invoke the Java Virtual Machine from Javascript). As explained in [34], installing the Java plug-in implies trusting Sun (the provider of Java, now part of Oracle), and the code produced by Sun. As a result, many people are unwilling to install Java plug-ins; some do not even enable Javascript.

Second, there is the issue of client performance. Helios requires expensive cryptographic computation in the client browser. In the UCL election, it is reported that it took on average 3-4 seconds to encrypt a vote [33]. During a public usability study of the Helios system, conducted at the University of Waterloo, encryption seemed to take longer – half of the participants (20 in total) received at least one script timeout message because encryption took too long [37]. Voters using old, or slow, computers may not be able to participate in the voting.

Third, the downloaded code presents another issue. In Helios, the client browser executes downloaded code from the server. Some IACR members were concerned with this, as one member indicates: "Once I started the voting system, I had no clue whether the applet running in my browser was actually the Helios voting application or some other application trying to mimic the Helios voting system" [34]. Malicious code can not only easily compromise vote secrecy, but also cause other harm to befall a user's computer [39].

Fourth, as acknowledged in the Helios paper [33], the management of tallying authorities proved to be a real challenge in practice. In the UCL election, the tallying authorities were selected from students and administrative staff etc. Many did not have a computer science background (so they were not from the same or related CS labs). In theory, the tallying authorities are responsible for keeping the private keys secret, and more importantly, keeping the keys available during the election (losing keys would cause DoS attacks to the whole election). This is beyond the comfort zone of ordinary people who are not computer experts and know little about cryptography. In reality, an election commission was organized with several computer experts. The election commission assisted in almost all the tasks – from buying new laptops, removing hard disk drives, disabling wireless network cards, to booting up from Linux CDs, running key generation code, storing the generated keys on USB sticks, and executing decryption code after voting. As noted in [33], this required the tallying authorities to "place significant trust in the election commission" to protect the private keys. Still, this was considered not good enough. The risk of having some USB sticks accidentally damaged by the tallying authorities was deemed high, so, in addition, UCL used a notary public to centrally maintain the backups of the all the private keys [33].

Motivated by the above practical issues, we chose a completely different approach to construct a cryptographic voting protocol. Our design strategy is to pre-compute almost all the cryptographic operations before an election, in such a structured way that the election will be self-tallying. The pre-computation provides two advantages: 1) it allows a thin client as the client does not need to do any computation at all[2], hence addressing the first three issues; 2) it permits self-tallying, so anyone can tally votes without needing tallying authorities, hence addressing the

[2]This is consistent with the recommendation in the IACR report [34] of moving client encryption to the server, but our solution involves a thorough and systematic change in the overall design.

fourth issue.

Our protocol is based on an innovative cancellation formula (details will be explained in Section III-B2), which was first presented by Hao and Zieliński in 2006 to solve the anonymous veto problem [14] and later applied by Hao, Ryan and Zieliński to solve the *decentralized* e-voting problem [12]. Our contributions in this paper are twofold. Applying the cancellation formula to *centralized* e-voting to assure tallying integrity is the primary contribution. Although the formula itself is not new (see [12], [14]), the use of it in centralized e-voting is new. The result is essentially a new voting protocol. Our secondary contribution is to effectively combine the cancellation formula [14], homomorphic encryption [12] and Benaloh's voter-initiated audit framework [27] into an overall efficient voting system. In the following section, we will explain in detail how the system works.

III. The DRE-i Protocol

In this section, we describe a cryptographic e-voting protocol called DRE-i, where i stands for integrity. For simplicity of discussion, we will mainly explain the protocol in the context of touch-screen based on-site voting, and later show its application in remote e-voting.

A. Integrity requirements

To ensure integrity, a voting protocol should fulfill the following requirements.

1) **Ballot well-formedness:** The ballot must have the correct format to represent exactly one vote.
2) **Recorded as cast:** The recorded vote must be the same as the one the voter intended to cast.
3) **Tallied as recorded:** The tally must be the same as the sum of the recorded votes.

These requirements are intuitive. The first requirement limits a single ballot to have only one vote. For example, in the single-candidate election, the ballot should contribute either 0 or 1 to the tally, nothing more than that. The Zero Knowledge Proof (ZKP) is a well-established technique to ensure ballot well-formedness [17], [18]. The second requirement states the machine must record the correct input from the voter. A widely adopted solution, which is also used in our protocol, is via Benaloh's voter-initiated auditing [27]. The third one is a crucial requirement [26]. Satisfying this requirement without involving tallying authorities is the main contribution of this paper. Additional requirements such as coercion resistance can be found in [6], [8], [9]. We will explain in Section IV that our protocol also fulfills those requirements.

B. Three Stages of Voting

The DRE-i protocol consists of three stages: ballot generation, ballot casting and ballot tallying. The following sections explain each stage in detail.

1) Ballot generation: Let G denote a finite cyclic group of prime order q in which the Decision Diffie-Hellman (DDH) problem is intractable [16]. Let g be a generator in G. The parameters (G, g) are publicly agreed before the election starts.

Let us first consider the single-candidate case. The system generates n ballots where n is significantly larger (say 10 times more) than the total number of the eligible voters. The extra ballots are used for auditing purposes.

For each ballot, the system computes a random public key g^{x_i}, where $x_i \in_R [1, q-1]$. When this is done for all the ballots, the system computes $g^{y_i} = \prod_{j<i} g^{x_j} / \prod_{j>i} g^{x_j}$ for every ballot. Here, we call g^{y_i} a restructured public key, because it is constructed by multiplying all the random public keys before i and dividing all the public keys after i. Given that x_i is random, $y_i \neq 0$ holds with an exceedingly overwhelming probability. (If $y_i = 0$, it would be publicly obvious that the machine is misbehaving.) In the following theorem, we assume the machine selects x_i properly at random and keep the values secret. In Section IV, we will discuss the implications if the machine deviates from this assumption.

Theorem 1. *Under the Decision Diffie-Hellman assumption, provided $y_i \neq 0$, the term $g^{x_i y_i}$ is indistinguishable from a random non-identity element in the group G.*

Proof: By the protocol definition, $x_i \in_R [1, q-1]$ and $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$. The y_i is random over Z_q and is unrelated to x_i. Since, $y_i \neq 0$, we have $y_i \in_R [1, q-1]$. To obtain a contradiction, we assume there is a polynomial-time algorithm (an oracle) to distinguish $g^{x_i \cdot y_i}$ from a random non-identity element in the group G. Without loss of generality, we only discuss the case that $i = 1$.

Given g^a, g^b, g^{ab} where $a, b \in_R [1, q-1]$, the DDH assumption states that g^{ab} is indistinguishable from a random non-identity element in G (see [16]). We now show how the assumed oracle can break this assumption. First, we do an efficient transformation as shown in Table I. Basically, we let $g^{x_1} = g^a$ and $g^{x_2} = g^{-b-\sum_{i>2} x_i}$ where $x_i \in_R [1, q-1]$ for $i > 2$. We do not need to know the values of x_1 and x_2. Following the definition of y_i, we obtain $y_1 = -\sum_{i>1} x_i = b$. Thus, $g^{x_1 y_1} = g^{ab}$ (the value was given). For $i = 2$, we can compute $g^{x_2 y_2} = g^{(-b-\sum_{i>2} x_i)(a-\sum_{i>2} x_i)}$. Similarly, we can easily compute $g^{x_3 x_4}$ until $g^{x_n y_n}$. As shown in Table I, the resultant new table is indistinguishable from the old one. Since the assumed oracle can efficiently distinguish $g^{x_1 x_2}$ from random, it thus can efficiently distinguish g^{ab} from random. This however contradicts the assumption that the DDH problem is intractable. The same argument applies if any $g^{x_i y_i}$ $(i \neq 1)$ is distinguishable from random. ■

The "Yes"/"No" value in each ballot is encoded in the form of as $C_i = g^{x_i y_i} \cdot g^{v_i}$ where $v_i = 0$ for "No" and 1 for "Yes". Theorem 1 shows that the no-vote, $g^{x_i y_i}$, is

$\mathbf{g^x}$	$\mathbf{g^y}$	$\mathbf{g^{xy}}$
g^{x_1}	g^{y_1}	$g^{x_1 y_1}$
g^{x_2}	g^{y_2}	$g^{x_2 y_2}$
g^{x_3}	g^{y_3}	$g^{x_3 y_3}$
...	...	...
g^{x_n}	g^{y_n}	$g^{x_n y_n}$

$\Rightarrow$

$\mathbf{g^x}$	$\mathbf{g^y}$	$\mathbf{g^{xy}}$
g^a	g^b	g^{ab}
$g^{-b-\sum_{i>2} x_i}$	$g^{a-\sum_{i>2} x_i}$	$g^{(-b-\sum_{i>2} x_i)(a-\sum_{i>2} x_i)}$
g^{x_3}	$g^{a-b-\sum_{i>2} x_i-\sum_{i>3} x_i}$	$g^{x_3(a-b-\sum_{i>2} x_i-\sum_{i>3} x_i)}$
...	...	...
g^{x_n}	g^{a-b-x_n}	$g^{x_n(a-b-x_n)}$

Table I

TABLE TRANSFORMATION. THE VALUES OF a, b ARE RANDOM OVER $[1, q-1]$. IN EITHER TABLE, THE x_i VALUES ARE RANDOMLY CHOSEN FROM $[1, q-1]$. CLEARLY, IN THE LEFT TABLE, THE EXPONENTS OF g^x ARE ALL RANDOM; IN THE RIGHT TABLE, THE EXPONENTS OF g^x ARE ALL RANDOM TOO. THE TWO TABLES ARE INDISTINGUISHABLE.

indistinguishable from random. Clearly, the yes-vote, $g^{x_i y_i} \cdot g$, is indistinguishable from random too. However, if both no-vote and yes-vote are published, the correlation between the two will make it trivially obvious which is "No" and which is "Yes".

In addition, the system needs to compute a 1-out-of-2 ZKP for each yes/no value. This is to ensure that the value of the vote is indeed in the correct form of $C_i = g^{x_i y_i} \cdot g^{v_i}$ where $v_i \in \{0, 1\}$. In other words, the value v_i can only be one of the two: 0 and 1. We adopt the standard 1-out-of-n ZKP technique (also known as the CDS technique) presented in [17]. Here, we use $n = 2$.

Given an ElGamal encryption $(x, y) = (g^{x_i}, h^{x_i} m)$, the CDS technique demonstrates that m is either m_0 or m_1 without revealing which[3]. This is achieved by proving the following OR statement:

$$\log_g x = \log_h (y/m_0) \quad \vee \quad \log_g x = \log_h (y/m_1)$$

Figure 2 shows a 3-move interactive protocol using the CDS technique, with $m_0 = g^0$, $m_1 = g^1$ and $h = g^{y_i}$. Applying the Fiat-Shamir's heuristics makes the protocol non-interactive [17], by letting $c = H(i, x, y, a_1, b_1, a_2, b_2)$ where H is a publicly secure hash function. In summary, the 1-out-of-2 ZKP for C_i contains: $(w, a_1, b_1, a_2, b_2, d_1, d_2,)$. Additional information on the 1-out-of-n Knowledge Proof can be found in [17], [18].

As shown in Table II, we define the cryptograms for the yes/no votes as follows. The cryptogram of the no-vote contains $g^{x_i y_i}$ and a 1-out-of-2 ZKP. Similarly, the cryptogram of the yes-vote comprises $g^{x_i y_i} \cdot g$ and a corresponding 1-out-of-2 ZKP. At the end of the ballot generation, the random public keys are published on the bulletin board, while the cryptograms are kept secret by the machine. At this stage, the x_i secret values become technically redundant and will not be needed for the rest of the protocol execution[4].

[3]In our case, the public key h equals g^{y_i} and is dynamically constructed by combining other public keys. We use the symbol h for simplicity. The context should make the meaning clear.

[4]From the protocol's perspective, the x_i values are no longer needed. However, in practice, there may be reasons to retain these values in the DRE during the course of election, as they form the most compact backup data. Note that leaking x_i is effectively equivalent to revealing the pre-computed cryptograms – in either case, the leakage will present itself to the public as clear evidence of the machine's misbehavior.

2) Ballot casting: While the ballot generation was performed before the election in a controlled environment (where party representatives can observe), ballot casting occurs at the polling stations on the election day. The environment at the field deployment of the DRE becomes more adverse. However, note that all the random values used in the computation of the cryptograms have been chosen before election and the random public keys have been published on the public bulletin board (see Table II). This greatly limits any room of maneuver by a DRE once it is deployed in the field. The ballot casting basically involves very simple operations to print out the pre-computed cryptograms depending on the voter's choice, as we explain below.

As before, we assume the eligible voter has been properly authenticated before entering the private voting booth and that the machine does not know the real identity of the voter. The voter presents the authentication token to the DRE machine and sees the same "select and confirm" interface on the touch screen (Figure 3). The ballot no i may be incremental or randomly assigned – there is no significant difference from the protocol's perspective. To cast the ballot, the voter follows the same two steps.

In step one, the voter selects a choice on the screen. Meanwhile, the machine prints the following commitment data on the paper: the ballot no i, the cryptogram of the selected choice (i.e., $g^{x_i y_i} \cdot g^{v_i}$ where $v_i = 0$ or 1 for "No"/"Yes" choice correspondingly, and a 1-out-of-2 Zero Knowledge Proof to prove that v_i is indeed one of the two values $\{0, 1\}$). The commitment transcript is digitally signed by the machine to prove the authenticity. The same content, including the digital signature, will be available on the bulletin board for public verification.

In step two, the voter either confirms or cancels the selection. If he chooses to confirm, the system will print a "finish" message on the paper. However, if the voter chooses to cancel, the DRE machine will print the selected choice, and reveal the other cryptogram onto the paper. The touch screen will return to the "select candidate" step. A voter is entitled to cast as many dummy votes as he wishes[5], but is allowed to cast only one valid vote. As in the previous

[5]Obviously, this is bounded by n and, in practice, a reasonable limit would be enforced.

Voter			Verifier
$v=1$	$v=0$		
$w, r_1, d_1 \in_R \mathbb{Z}_q$	$w, r_2, d_2 \in_R \mathbb{Z}_q$		
$x \leftarrow g^{x_j}$	$x \leftarrow g^{x_j}$		
$y \leftarrow h^{x_j} \cdot g$	$y \leftarrow h^{x_j}$		
$a_1 \leftarrow g^{r_1} x^{d_1}$	$a_1 \leftarrow g^{w}$		
$b_1 \leftarrow h^{r_1} y^{d_1}$	$b_1 \leftarrow h^{w}$		
$a_2 \leftarrow g^{w}$	$a_2 \leftarrow g^{r_2} x^{d_2}$		
$b_2 \leftarrow h^{w}$	$b_2 \leftarrow h^{r_2}(y/g)^{d_2}$	$\xrightarrow{x, y, a_1, b_1, a_2, b_2}$	
$d_2 \leftarrow c - d_1$	$d_1 \leftarrow c - d_2$	$\xleftarrow{c}$	$c \in_R \mathbb{Z}_q$
$r_2 \leftarrow w - x_j d_2$	$r_1 \leftarrow w - x_j d_1$	$\xrightarrow{d_1, d_2, r_1, r_2}$	$c \stackrel{?}{=} d_1 + d_2$
			$a_1 \stackrel{?}{=} g^{r_1} x^{d_1}$
			$b_1 \stackrel{?}{=} h^{r_1} y^{d_1}$
			$a_2 \stackrel{?}{=} g^{r_2} x^{d_2}$
			$b_2 \stackrel{?}{=} h^{r_2}(y/g)^{d_2}$

Figure 2. Proof of Validity: the ballot (x, y) is either $(g^{x_j}, h^{x_j} \cdot g)$ or (g^{x_j}, h^{x_j}) where $h = g^{y_j}$.

Ballot No	Random public key	Restructured public key	Cryptogram of no-vote	Cryptogram of yes-vote
1	g^{x_1}	g^{y_1}	$g^{x_1 \cdot y_1}$, 1-of-2 ZKP	$g^{x_1 \cdot y_1} \cdot g$, 1-of-2 ZKP
2	g^{x_2}	g^{y_2}	$g^{x_2 \cdot y_2}$, 1-of-2 ZKP	$g^{x_2 \cdot y_2} \cdot g$, 1-of-2 ZKP
...	...	...	...	...
n	g^{x_n}	g^{y_n}	$g^{x_n \cdot y_n}$, 1-of-2 ZKP	$g^{x_n \cdot y_n} \cdot g$, 1-of-2 ZKP

Table II
BALLOT GENERATION. THE TABLE, EXCEPT THE LAST TWO COLUMNS, IS PUBLISHED ON A PUBLIC BULLETIN BOARD BEFORE THE ELECTION STARTS.

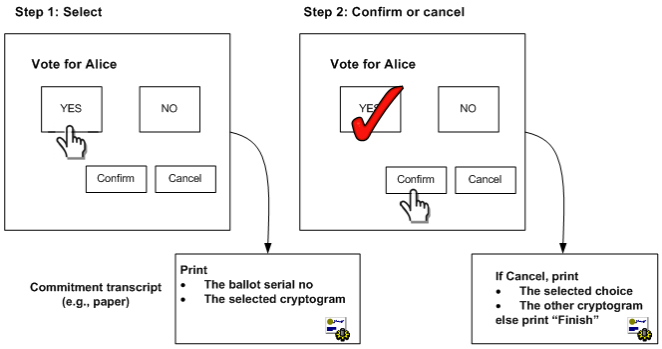

Figure 3. A DRE with integrity (DRE-i) voting system . A confirmed ballot is termed a "valid" vote while a canceled one is referred to as a "dummy" vote.

step, the commitment transcript is digitally signed; the same content will also be available on the bulletin board.

The cancel option serves for auditing. Note that the 1-out-of-2 ZKP ensures that the formats of "No"/"Yes" votes are in the form of $g^{x_i y_i} \cdot g^{v_i}$, $v_i \in \{0, 1\}$, but it does not guarantee the correct assignment of "0"/"1" to "No"/"Yes". The voter-initiated auditing addresses this (the same auditing idea was first proposed by Benaloh [27]). Auditing can be performed by any voter during any stage of the election. When all the voters have cast their votes, the system will reveal the remaining ballots as "dummy" and publish them on the public bulletin board (displayed as if canceled by the voters).

The paper receipt for ballot i contains the printed data from both steps. The voter is free to take home the receipt and verify it against the public bulletin board that his vote has been indeed included. The receipt does not reveal whom the voter has voted for, therefore preventing potential coercion and voter-buying.

3) Ballot tallying: Tallying the ballots is a case of multiplying the published cryptogram V_i (for dummy votes, only the *no*-value) all together (See Table III). Thus, we have $\prod_i V_i = \prod_i g^{x_i y_i} g^{v_i} = \prod_i g^{v_i} = g^{\sum_i v_i}$. The key to the tallying process is the fact that $\sum x_i y_i = 0$, which we refer to as the "cancelation formula" (see Proposition 1. The formula appeared earlier in [12], [14].). The term $\sum_i v_i$ is the total number of the "yes" votes. Since it is a relatively small number, it is feasible to compute it by exhaustive search. However, this exhaustive search is not entirely necessary. Since the machine records the ballots directly, it can announce the count of "yes" votes, β, right after the election. Everyone can verify whether g^{β} and $g^{\sum_i v_i}$ are equal. This takes only one exponentiation. Also, everyone can count the number of dummy votes from the bulletin board, which we denote as λ. Thus, the tally of "no" votes is $\alpha = n - \beta - \lambda$.

No i	Random pub key g^{x_i}	Restructured pub key g^{y_i}	Published Votes V_i	ZKPs
1	g^{x_1}	g^{y_1}	Valid: $g^{x_1 \cdot y_1}$	a 1-of-2 ZKP
2	g^{x_2}	g^{y_2}	Valid: $g^{x_2 \cdot y_2} \cdot g$	a 1-of-2 ZKP
3	g^{x_3}	g^{y_3}	Dummy: $g^{x_3 \cdot y_3}$, $g^{x_3 \cdot y_3} \cdot g$	two 1-of-2 ZKPs
...	...	...	...	...
n	g^{x_n}	g^{y_n}	Dummy: $g^{x_n \cdot y_n}$, $g^{x_n \cdot y_n} \cdot g$	two 1-of-2 ZKPs

Table III

BALLOT TALLYING. THIS ENTIRE TABLE IS PUBLISHED ON THE PUBLIC BULLETIN BOARD. A VOTE CAN BE EITHER VALID OR DUMMY. BALLOT NO. 1 SHOWS AN EXAMPLE OF A VALID "NO" VOTE, AND NO. 2 SHOWS AN EXAMPLE OF A VALID "YES" VOTE. TALLYING IS TO MULTIPLY ALL THE V_i VALUES (ONLY INCLUDING THE "NO" VOTES FOR THE DUMMY CASE).

As we will further explain in Section IV-E, the tallying process must admit all the cast votes, including the dummy ones. Partial tallying will not lead to any meaningful result – in other words, "every vote must count".

Proposition 1 (Cancellation forumla)**.** *For the x_i and y_i as defined in the protocol, $\sum_i x_i y_i = 0$.*

Proof: By definition $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$, hence

$$\begin{aligned}\sum_i x_i y_i &= \sum_i \sum_{j<i} x_i x_j - \sum_i \sum_{j>i} x_i x_j \\ &= \sum_{j<i} \sum x_i x_j - \sum_{i<j} \sum x_i x_j \\ &= \sum_{j<i} \sum x_i x_j - \sum_{j<i} \sum x_j x_i \\ &= 0.\end{aligned}$$

■

C. Extension to multiple candidates

There are several ways to extend a single-candidate election to multiple candidates [12]. A preferred method is attributed to Cramer *et al.* [18]: suppose that we have n votes, choose m so that m is the smallest integer such that $2^m > n$. Now the vote for candidate 1 is encoded as 2^0, for candidate 2 as 2^m, for candidate 3 is 2^{2m}, and so on. In other words, redefine the encoding value v_i within the cryptogram definition $C_i = g^{x_i y_i} \cdot g^{v_i}$ as:

$$v_i = \begin{cases} 2^0 & \text{if vote for candidate 1} \\ 2^m & \text{if vote for candidate 2} \\ \dots & \dots \\ 2^{(k-1)m} & \text{if vote for candidate } k \end{cases}$$

Tabulation is much as before: $\prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$. The votes are summed and the super-increasing nature of the encoding ensures that the total can unambiguously be resolved into the totals for the candidates. Hence, $\sum_i v_i = 2^0 \cdot c_1 + 2^m \cdot c_2 + \ldots + 2^{(k-1)m} \cdot c_k$, where c_1 to c_k are the counts of votes for the k candidates correspondingly. As before, the machine will announce the counts of votes right after the election. Anyone can verify the counts against $g^{\sum_i v_i}$, which takes a single exponentiation.

D. Voter privacy

First, we should acknowledge that when it comes to voter privacy, there is an inherent limitation with the DRE. A voter's privacy is constrained by the necessary human interaction with the machine, which records the votes directly. A corrupted touch screen terminal can easily learn the secret choice, just as a corrupted web browser can easily disclose the secret vote in the Helios system (see [39]).

Still, it is important to protect voter privacy. Our solution is to depend on procedural means to keep voters anonymous. For example, as we stated earlier, the voting officials shall ensure: 1) the voting booth is private; 2) the DRE machine does not know the voter's real identity; and 3) the published ballots do not show any linkage to the voters. These measures serve to decouple the voter's real identity from each cast ballot, and hence to preserve the voter's privacy.

E. Remote e-voting

We now take a broad interpretation of the DRE voting system: which not only includes on-site touch-screen machines, but also remote voting systems conducted via the Internet or mobile phones. In all cases, the system records the votes directly, although the security environments are different.

The DRE-i protocol is generically applicable to both on-site and remote e-voting scenarios. The protocol remains basically the same although the implementations are quite different. For example, in web-based Internet voting, the DRE machine may commit data by sending a signed email (as opposed to printing on paper). Similarly, if a mobile phone is used to vote, a signed Short Message Service (SMS) may be sent.

However, we need to stress that on-site and remote voting applications have distinct voting environments, each with an impact on security. Most notably, in a remote setting, we will lose effective procedural and physical protections that are available in an on-site election. Consequently, it becomes much harder to keep voters anonymous. In the UCL election, the voters were anonymized and only the aliases were published [33]. However, this only provides a weak layer of protection; a corrupted server might be able to match the aliases to real names. In an electronic world, where transmitted messages leave traces in the network log

files, maintaining real anonymity is a challenging research program in itself.

The loss of physical and procedural protections also opens up a number of new attacks – for example, a voter may be cajoled in disclosing their vote via a "bogus" website; the actual vote may be conducted under the duress of a coercer; voting credentials may become an item of profitable trade et cetera. Also, any independent observation of "counting valid voters" at the polling station will not be possible. Nevertheless, remote e-voting may still prove useful in some specifically identified scenarios, where the concerns on coercion and voter privacy are low – for example in the UCL [33] and IACR elections [34].

IV. DRE-i Analysis

Electronic voting is a complex problem. It requires more than just technical considerations as it crosses the disciplines of politics, human psychology, security economics and sociology et cetera [1]. Within this paper we only focused on solving a technical problem: tallying integrity. However, it is unlikely that any single technical protocol alone can guarantee a secure election. Correct implementation of the protocol is crucial. Also, there are realistic threats that our protocol alone cannot address.

In the following sections, we perform a comprehensive analysis of the proposed DRE-i protocol: explaining the technical properties of the protocol, discussing protocol implementations, highlighting practical threats concerning deployment as well as suggesting possible mitigation strategies.

A. Technical Properties of the Protocol

The DRE-i protocol fulfils the three integrity requirements as described in Section III-A. The use of the CDS technique (i.e., the 1-out-of-n zero knowledge proof) ensures the correct format of the ballot [17], thus fulfilling the first requirement. The second requirement is satisfied by the auditing function as described in [27]. Any voter can be an auditor by simply pressing the "cancel" button. The third requirement is fulfilled by the use of an innovative "cancellation forumla" [14] together with homomorphic encryption [12]. This permits anyone to easily verify the tally, based on the encrypted data displayed on the public bulletin board, without relying on any tallying authority.

In addition, the protocol protects the secrecy of the valid votes. The published value for a valid vote, $g^{x_i y_i} \cdot g^{v_i}$ for $v_i = 0$ or 1, is indistinguishable from random (see Theorem 1) and the associated 1-out-of-2 ZKP reveals nothing more than the statement: the v_i is either 0 or 1 [17]. If the vote is dummy, both cryptograms will be revealed. A dummy vote requires no secrecy since it does not add to the tally. As with any DRE system, the machine naturally learns the value of each vote (i.e., "Yes" or "No" for a single-candidate election). Our protocol cannot prevent a corrupted machine leaking the secret values. However, there are non-technical measures to further protect the voter's privacy – for example, through procedural means to assure voters that voting is anonymous.

The paper receipt in our protocol is coercion free. As we detailed earlier, if the voter chooses to confirm the vote, the receipt does not leak any information about whom he had voted for. This prevents potential coercion and vote-buying. If, however, the voter opts to cancel the vote, the receipt will reveal the selected choice, but the vote has been declared dummy. A dummy vote is useless to the coercer.

B. Estimating the Computation Cost

We begin by examining ballot generation. This stage involves computationally intensive operations. For a typical scenario, let us assume $n = 10^5$ (which is 10 thousand voters at a polling station times a safety factor of 10 for auditing). Also, we assume a typical cyclic group setting where p is 1024-bit and q is 160-bit.

As shown in Table II, we need to compute g^{y_i} for each ballot. At first glance, this is very expensive, taking approximately $n = 10^5$ multiplications to compute g^{y_1}. However, note that $g^{y_2} = g^{y_1} \cdot g^{x_2} \cdot g^{x_1}$. More generally, $g^{y_i} = g^{y_{i-1}} \cdot g^{x_i} \cdot g^{x_{i-1}}$ for $i > 1$. Thus, computing g^{y_i}, for $i = 2, 3, \ldots, n$, incurs negligible cost.

For each ballot i, exponentiation is the predominant cost factor. It takes one exponentiation to compute g^{x_i}, one to compute $g^{x_i y_i}$ and four to compute the 1-out-2 ZKP for each no/yes vote, totalling ten exponentiations. Each exponentiation takes approximately 5 milliseconds on a 2.33-GHz MacBook laptop [13]. Therefore, pre-computing all ballots on a single laptop takes $0.005 \times 10 \times n = 5 \times 10^3$ seconds = 1.4 hours. (We have not factored in the use of any form of optimization technique e.g. caching.)

In the ballot casting stage, the computational cost incurred by the DRE machine is very small – the machine merely needs to print out the pre-computed cryptogram according to the voter's choice. The main delay is most likely caused by printing.

The ballot tallying involves multiplying n group elements to obtain $g^{\sum v_i}$. One exponentiation requires an average of $1.5 \times \log_2 q = 240$ multiplications. The multiplication will take approximately $n/240 \times 0.005 = 2.08$ seconds. Verifying the tally against the count accounted for by the DRE requires one additional exponentiation: that is, another 0.005 seconds.

In addition, before an election, anyone can verify that the published random public keys g^{x_i} lie within the prime-order group, are randomly distributed and that the values of g^{y_i} are correctly computed. To verify the ZKP for the published vote V_i, it is necessary to first validate the order of V_i. This requires an exponentiation (for both the valid and dummy cases); it takes a further four exponentiations (using the simultaneous computation technique [38] to compute a_1, a_2, b_1, b_2) to verify the 1-out-of-2 ZKP as shown in Figure 2.

In total, it takes roughly 0.025 seconds to verify a ZKP on a laptop.

In summary, for the example of 100,000 votes, ballot generation will take about 1.4 hours on a laptop. During ballot casting, the computational cost only involves a digital signing; the main delay will be the time it takes to print the receipt. Finally, in takes approximately 2 seconds to verify the tallying. The verification of the published ZKPs will be a distributed effort, which takes 2,500 seconds in total. These estimates suggest the feasibility of our system.

C. National versus Regional Level

For the simplicity of illustration, we have so far used the same DRE in both ballot generation and ballot casting. This is, however, not the most cost-effective way to implement the system. In practice, we could employ a number of fast servers, in a controlled environment, to generate all cryptograms and then distribute to the DREs. Since the computational load on the DREs during the ballot casting is very low, it is possible to build those DREs using low-cost hardware. These are, however, implementation details.

Another implementation option concerns whether to position the DRE-i protocol at the national or regional level. In terms of the former, a central DRE system generates the ballots for the entire population and then distributes portions of the ballots to the DRE machines at each region (i.e., precinct). The tallying will be over the entire population, hence not revealing any partial tallying information at the regional level. In the latter, each region manages its own DRE system and performs independent tallying. The regional tallied results are then summed to form the total tally of the nation. The second approach is more manageable, and is in accord with widely adopted existing practice [29].

D. Subliminal Channels

A subliminal channel is concerned with information leakage about a cast ballot to a third-party. Karlof, Sastry and Wagner describe possible subliminal channel attacks against cryptographic e-voting protocols such as Neff's and Chaum's [9]. The threat of subliminal channels also applies to the Helios voting system [39]. The DRE-i protocol is no exception, as we explain below.

During the ballot generation, we have assumed that the DRE machine generates the x_i values randomly and secretly. However, a corrupted machine may deviate from this assumption. It may choose the x_i values from a low-entropy source, so the value $g^{x_i y_i}$ will no longer be indistinguishable from random. In this case, the public keys g^{x_i} will not be randomly distributed and, therefore, subject to detection. Hence, more likely, the corrupted machine will choose x_i at random, but leak the values to the coercer. This is an inherent threat with any DRE-based voting system due to the machine directly recording the vote and learning the voter's secret choice. After the election, all the x_i values together with the unused cryptograms should be permanently deleted. In any event, if any of these values becomes known to the public, this will present itself as clear evidence about the machine's misbehavior.

E. Denial of Service

The electronic data published by the DRE machine must be precise. If a single bit of the vote gets flipped, it will break the well-formedness of the vote, thus bringing it to public attention. Also, all votes must be included within the tallying process. If a single vote goes missing, rather than going unnoticed, the tallying process at the particular precinct will fail – essentially, a publicly evident Denial of Service attack. This may appear, at first glance, as a weakness. On the contrary, this is a strength of the protocol, ensuring every vote is counted. This effectively demands that DRE vendors must follow stringent engineering practice to ensure hardware and software robustness (which is generally required in most security-critical systems [1], e.g., payment solutions in the banking industry).

F. More Powerful Coercion

We previously highlighted that the DRE-i receipt is coercion free. However, there may be more powerful coercion scenarios: e.g., threatening voters not to vote at all, forcing voters to film everything they do in the private booth, or colluding with the DRE machine to discover the voters' secret votes et cetera [9]. It is beyond our protocol to address these threats. If such powerful coercion becomes wide spread, the value of the election itself may be called into question.

G. Voter Enrollment and Authentication

Voter enrollment and authentication are two important pre-conditions for our protocol. The former helps to determine the volume of ballots to generate. The latter is crucial to ensure *one-man-one-vote*. The election staff at the polling station must keep a reliable record of how many voters have voted on the election day. This number can then be compared with the total count of valid votes published on the public bulletin board. It is possible, that once inside the booth, the voter fails to cast a vote, or only casts dummy votes. This would be evident by the paper receipt or the returned authentication token.

H. Social engineering attacks

Typically, threats have centered on how to identify a misbehaving DRE machine, however, we also need to consider the case where the DRE is honest, but the voter is misbehaving. We call this a "social engineering attack", as the attack is not technical in nature, but can sometimes be very effective if countermeasures are lacking. We highlight a few attacks below. Note that they are also applicable to other e-voting protocols in general [25], [26], [32], [33].

As an example, suppose the voter selects "yes" in the first step, and then chooses to cancel. The machine dutifully prints a receipt to reveal the "yes" selection and declares the vote as dummy. The voter may now report to an election staff that he actually selected "no". We assume the purpose of this attack is to discredit the machine.

In such a situation, is the voter misbehaving or the machine? For the election staff, it is not easy to tell the difference. If the voter raises the dispute, one practical resolution is to invite several independent observers to supervise the voting. The independent observers do not need to learn the voter's secret. For example, the voter casts several dummy votes under independent observation until being happy that the machine is acting in accordance with his wishes.

In another example, the voter might dispute that he chose "cancel", but the screen displays "your vote has been confirmed". Again, there is no easy way to tell who is lying. One resolution may be to have several voting officials jointly agree to mark the ballot as "disputed" and allow the voter to cast another vote. When the election finishes, the "disputed" ballots, together with the unused ballots, will be revealed by the DRE and declared "dummy". (Essentially, this is the same as allowing re-voting as in the UCL election [33].) The key to resolution is to ensure that the handling of the "disputed" votes is transparent to the public. In any event, the total number of the valid votes published on the public bulletin board shall match the number of the voters who actually cast their votes on the election day.

I. Receipt Verification

The generated paper receipt provides the voter the ability to verify whether his vote has been correctly included in the final tally. The receipt itself does not reveal any information about whom the voter has voted for. Still, it is important for voters to verify the receipts, so that DRE fraud, if any, can be detected.

However, in general, we should assume that many voters will not endeavour to verify the receipts by themselves. This is a general problem for many cryptographic e-voting protocols [25], [26], [33], and not specific to our protocol. It is therefore crucial for the election officials to establish incentive schemes, that encourage verification of receipts, and to provide voters with all the necessary facilities and assistance to do so (say near the exit of the polling station).

In fact, we have designed the protocol in such a way that the voter does not have to understand cryptography in order to verify the receipt. During the voting, if he opts to "cancel" the vote, he only needs to verify that the revealed candidate name on the receipt is the same as he chose in the first step (see Figure 3); if not, he should immediately raise a dispute. After voting, he merely needs to verify that – possibly with the assistance of election officials near the exit – the receipt is indeed published on the bulletin board. He does not need to verify the receipt by himself. As long as the same content is available on the public bulletin board, anyone with the knowledge and skill can write a program to verify the receipts in a batch.

In practice, there may be dedicated auditors from different election parties. They can choose to audit the machine, by casting dummy votes, at any time during the election. They may even have the expertise to verify the receipts by themselves. However, we should note that dedicated auditors cannot replace the general public in auditing. For ordinary voters, the ability to audit the system is important to build up public trust in the election.

V. Conclusion

In this paper, we first reviewed the practical deployments of DREs, which had seen a rush of adoption in 2000 followed by a wave of rejection in 2006. This shows that DRE technology is still relatively young. We then studied the Helios e-voting system, which presents a particularly interesting case to examine the conflicts between theory and practice. Some of the theoretical assumptions – such as crypto-aware users, a trustworthy client execution environment and trustworthy tallying authorities – have been commonly made in the past literature but were challenged in practical applications.

Motivated to address the practical issues, we proposed the DRE-i protocol. The protocol is based on an innovative cancellation formula, combined with a pre-computation strategy; it adds strong assurance of tallying integrity to the DRE system, without altering the voter's intuitive voting experience; the auditing is voter-initiated and has been seamlessly integrated into the natural voting process; the protocol is generically applicable to both on-site and remote e-voting; the election is self-tallying, so the public can tally the votes without relying on trusted computing or tallying authorities; though the protocol primarily focuses on the tallying integrity, we also described procedural means to preserve the voter's privacy.

The integrity of an election underlies the integrity of democracy. With the DRE-i protocol, "every vote counts" is no longer a mere slogan.

Acknowledgement

This paper is built upon the previous published works authored in collaboration with Dr. Piotr Zieliński and Prof. Peter Ryan. They also made contributions into this paper. We also like to thank Ross Anderson, Joseph Bonneau and other members of the security group at the Computer Laboratory, University of Cambridge, for many useful comments.

References

[1] R.J. Anderson, *Security Engineering : A Guide to Building Dependable Distributed Systems*, Second Edition, New York, Wiley 2008.

[2] T. Kohno, A. Stubblefield, A.D. Rubin, and D.S. Wallach, "Analysis of an Electronic Voting System," Proceedings of the 25th IEEE Symposium on Security and Privacy, May, 2004.

[3] R. W. Gardner, S. Garera, A.D. Rubin, "Designing for Audit: A Voting Machine with a Tiny TCB," Proceedings of the 14th Financial Cryptography and Date Security, January, 2010.

[4] S. Garera and A.D. Rubin, "An Independent Audit Framework for Software Dependent Voting Systems," Proceedings of the 14th ACM Conference on Computer and Communications Security, October 2007.

[5] R.L. Rivest, W.D. Smith, "Three Voting Protocols: ThreeBallot, VAV, and Twin," Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, 2007.

[6] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. Rivest, P. Ryan, E. Shen, A. Sherman, "Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems Using Invisible Ink Confirmation Codes," Proceedings of the USENIX/ACCURATE Electronic Voting Workshop, 2008.

[7] D. Chaum, P.Y. Ryan, and S.A. Schneider, "A Practical, Voterverifiable, Election Scheme," ESORICS'05, LNCS 3679, pp. 118-139, 2005.

[8] D.R. Sandler, K. Derr, and D.S. Wallach, "VoteBox: A Tamper-Evident, Verifiable Electronic Voting System," Proceedings of the 17th USENIX Security Symposium, July 2008.

[9] C. Karlof, N. Sastry, D. Wagner, "Cryptographic Voting Protocols: A Systems Perspective," Proceedings of the 14th USENIX Security Symposium, pp. 33-50, August, 2005.

[10] R.A. Fink, A.T. Sherman, R. Carback, "TPM Meets DRE: Reducing the Trust Base for Electronic Voting Using Trusted Platform Modules," *IEEE Transactions on Information Forensics and Security*, No. 4. Issue. 4, pp. 628-637, 2009.

[11] A.J. Feldman, J.A. Halderman, E.W. Felten, "Security Analysis of the Diebold AccuVote-TS Voting Machine," Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, 2007.

[12] F. Hao, P. Ryan, P. Zieliński, "Anonymous Voting by 2-Round Public Discussion", *IET Information Security*, in press, 2010.

[13] F. Hao, P. Y. A. Ryan, "Password Authenticated Key Exchange by Juggling," Proceedings of the 16th International Workshop on Security Protocols, Cambridge, UK, April 2008.

[14] F. Hao, P. Zieliński, "A 2-Round Anonymous Veto Protocol," Proceedings of the 14th International Workshop on Security Protocols, Cambridge, UK, 2006.

[15] F. Hao, P. Zieliński, "The Power of Anonymous Veto in Public Discussion," *Sprigner Transactions on Computational Sciences IV*, pp. 41-52, 2009

[16] D. Stinson, *Cryptography: Theory and Practice*, Third Edition, Chapman & Hall/CRC, 2006.

[17] R. Cramer, I. Damgård, B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols," Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, LNCS, vol. 839, pp. 174-187, 1994.

[18] R. Cramer, M. Franklin, B. Schoenmakers and Moti Yung, "Multi-Authority Secret-Ballot Elections with Linear Work," EUROCRYPT '96, LNCS, vol. 1070, pp. 72-83, 1996.

[19] A. Kiayias, M. Yung, "Self-tallying elections and perfect ballot secrecy," Public Key Cryptography '02, LNCS, vol. 2274, pp. 141-158, 2002.

[20] J. Groth, "Efficient maximal privacy in boardroom votisng and anonymous broadcast," Financial Cryptography '04, LNCS, vol. 3110, pp. 90-104, 2004.

[21] D. Chaum, "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65-67, 1988.

[22] D. Chaum, "Untraceable Electronic Email, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, Vol. 24, No. 2, pp. 84–88, 1981.

[23] A.M. Keller, D. Mertz, J.L. Hall, A. Urken, "Privacy issues in an electronic voting machine," Proceedings of the 2004 ACM workshop on Privacy in the electronic society, pp. 33-34, 2004.

[24] A.T. Sherman, A. Gangopadhyay, S.H. Holden, G. Karabatis, A.G. Koru, C.M. Law, D.F. Norris, J. Pinkston, A. Sears, and D. Zhang, "An Examination of Vote Verification Technologies: Findings and Experiences from the Maryland Study," Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop, 2006.

[25] D. Chaum, "Secret-Ballot Receipts: True Voter-Verifiable Elections," *IEEE Security and Privacy*, vol. 2, no. 1, pp. 38-47, Jan. 2004.

[26] C.A. Neff, "A Verifiable Secret Shuffle and Its Application to E-Voting,"Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 116-125, 2001.

[27] J. Benaloh, "Ballot Casting Assurance via Voter-Initiated Poll Station Auditing," Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, 2007.

[28] R. Mercuri, "Electronic Vote Tabulation Checks & Balances," Ph.D thesis, University of Pennsylvania, 2000.

[29] G.C. Edwards III, "The 2000 U.S. Presidential Election," Taiwan Journal of Democracy, Vol. 2, N. 1, pp. 37-50, 2006.

[30] The commerical Syctl voting solution website: http://www.scytl.com

[31] R.L. Rivest and J.P. Wack, "On the notion of Software Independence in Voting Systems," 2006. Available at http://vote.nist.gov/SI-in-voting.pdf

[32] B. Adida, "Helios: web-based open-audit voting," Proceedings of the 17th conference on Security symposium, pp. 335-348, 2008.

[33] B. Adida, O. de Marneffe, O. Pereira, and J.J. Quisquater, Proceedings of the Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, 2009.

[34] S. Haber, J. Benaloh, S. Halevi, "The Helios e-Voting Demo for the IACR," June, 2010. Available at http://www.iacr.org/elections/eVoting/heliosDemo.pdf

[35] J. Blanc, "Challenging the Norms and Standards of Election Administration: Electronic Voting," Chapter of International Foundation For Electoral Systems (IFES) report, pp. 11-19, 2007

[36] "Help America Vote Act at 6", USA Today, October 29, 2008. http://www.usatoday.com/news/politics/election2008/2008-10-28-votingequipment_N.htm#table.

[37] J.L. Weber, U. Hengartner, "Usability Study of the Open Audit Voting System Helios," 2009. Available at www.jannaweber.com/wp-content/uploads/2009/09/858Helios.pdf

[38] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1996.

[39] S. Estehghari, Y. Desmedt "Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example," Proceedings of Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EV/WOTE'10), 2010.