Computational Election Verifiability: Definitions and an Analysis of Helios and JCJ

Ben Smyth
Huawei Technologies Co. Ltd.,
Boulogne-Billancourt, France
research@bensmyth.com

Steven Frink
Cornell University
Ithaca, NY, US
sfrink@cs.cornell.edu

Michael R. Clarkson
Cornell University
Ithaca, NY, US
clarkson@cs.cornell.edu

***Abstract*—Definitions of election verifiability in the computational model of cryptography are proposed. The definitions formalize notions of voters verifying their own votes, auditors verifying the tally of votes, and auditors verifying that only eligible voters vote. The Helios (Adida et al., 2009) and JCJ (Juels et al., 2010) election schemes are shown to satisfy these definitions. Two previous definitions (Juels et al., 2010; Cortier et al., 2014) are shown to permit election schemes vulnerable to attacks, whereas the new definitions prohibit those schemes.**

I. Introduction

Electronic voting systems that have been deployed in real-world, large-scale public elections place extensive trust in software and hardware. Unfortunately, instead of being trustworthy, many systems are vulnerable to attacks that could bring election outcomes into disrepute [18], [45], [54], [83]. So relying solely on trust in voting systems is unwise; verification of election outcomes is essential.[1]

Election verifiability enables voters and auditors to establish the correctness of election outcomes, regardless of whether the software and hardware of the voting system are trusted [1], [2], [24], [55], [74]. According to Kremer et al. [61], election verifiability encompasses three aspects:[2]

- *Individual verifiability:* voters can check that their own ballots are recorded.
- *Universal verifiability:* anyone can check that the tally of recorded ballots is computed properly.
- *Eligibility verifiability:* anyone can check that each tallied vote was cast by an authorized voter.

In this paper, we propose new definitions of these three aspects of verifiability in the computational model of cryptography. Because some electronic voting systems outsource voter authentication to third parties, whereas other voting systems implement it themselves, we give two variants of our definitions—one for systems with *external authentication* and another for systems with *internal authentication*.

We employ our definitions to analyze the verifiability of two well-known election schemes, Helios [5] and JCJ [57]. Helios is a web-based voting system that has been deployed in the real-world. JCJ is an election scheme that achieves *coercion resistance* and has been implemented as Civitas [28]. The Helios 2.0 election scheme is known to have vulnerabilities that enable attacks on verifiability, and several patches for those vulnerabilities have been proposed [16], [17], [33], [34]. By employing those proposed patches, we obtain a scheme called Helios 4.0 that satisfies our definition of election verifiability with external authentication. Helios 2.0, as expected, fails to satisfy our definition. Although a model of Helios 2.0 has been proved to satisfy a different definition of verifiability [69], that model uses an abstraction of cryptographic primitives that is insufficient to detect some vulnerabilities. Our model includes sufficient detail about cryptographic primitives to detect the vulnerabilities.

Helios-C [32], a variant of Helios in which ballots are digitally signed, satisfies our definition of election verifiability with internal authentication. Although we might expect JCJ to also satisfy this definition, it does not: a weakening of the definition is required, because a fully corrupt tallier could cause unauthorized votes to be tallied in JCJ. Civitas would require the same weakening.

Our definitions of election verifiability improve upon two previous definitions [32], [57] by detecting a new class of *collusion attacks*, in which the tallying procedure announces an incorrect tally, and the verification procedure colludes with the tallying procedure to accept the incorrect tally. Examples of collusion attacks include ballot stuffing, and announcing tallies that are independent of the election. Our definitions also improve upon those previous definitions by detecting a new class of *biasing attacks*, in which the verification procedure rejects some legitimate election outcomes. Examples of biasing attacks include rejecting outcomes in which a particular candidate does not win, and rejecting all election outcomes, even correct outcomes. And our definitions of election verifiability are more formal than existing work on *global verifiability* [66], [67], [69]. Global verifiability is parameterized on *goals*, which prior work did not formalize, as we discuss in Section VII.

This paper thus contributes to the security of electronic voting systems by

- proposing computational definitions of election verifiability,
- proving that well-known election schemes do (or do not) satisfy election verifiability, and
- identifying collusion and biasing attacks as new classes of attacks on voting systems and demonstrating that they

[1]*Doveryai, no proveryai* (trust, but verify) says the Russian proverb.
[2]This decomposition has been criticized [67]; we discuss it in Section VII.

are not detected by two earlier definitions.

Ours are the first proofs that Helios 4.0 and JCJ satisfy a computational definition of verifiability.

We proceed as follows. Section II defines election verifiability with external authentication. Section III analyzes Helios. Section IV defines election verifiability with internal authentication. Section V analyzes JCJ. Section VI shows that two previous definitions of election verifiability fail to detect collusion and biasing attacks. Section VII reviews related work, and Section VIII concludes. Proofs appear in the forthcoming technical report.

II. External Authentication

Some election schemes do not implement authentication themselves, but instead rely on an external authentication mechanism. Helios, for example, supports authentication with Facebook, Google and Yahoo credentials.[3] In essence, the election scheme outsources ballot authentication. We begin by defining election verifiability for that model.

A. Election scheme

An *election scheme with external authentication*, which henceforth in this section we abbreviate as "election scheme," is a tuple $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ of probabilistic polynomial-time (PPT) algorithms:

- **Setup**, denoted[4] $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$, is executed by the *tallier*, who is responsible for tallying ballots. Setup takes a security parameter k as input and outputs a key pair $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$, a maximum number of ballots m_B, and a maximum number of candidates m_C.[5]
- **Vote**, denoted $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$, is executed by voters. A voter makes a *choice* of candidate from a sequence $c_1, \ldots, c_{n_C}$ of candidates. A *well-formed* choice is an integer β, such that $1 \leq \beta \leq n_C$. Vote takes as input the public key $PK_{\mathcal{T}}$ of the tallier, the number n_C of candidates, the voter's choice β of candidate, and security parameter k. It outputs a ballot b, or error symbol $\perp$. An error might occur if the candidate choice is not well-formed or for other reasons particular to the election scheme.
- **Tally**, denoted $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k)$, is executed by the tallier. It involves a public *bulletin board* BB, which we model as a set.[6] Tally takes as input the public key $PK_{\mathcal{T}}$ and private key $SK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, and security parameter k. It outputs a tally $\mathbf{X}$ and a non-interactive proof P that the tally is correct. A *tally* is a vector $\mathbf{X}$ of length n_C such that $\mathbf{X}[j]$ indicates the number of votes for candidate c_j.[7]
- **Verify**, denoted $v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$, can be executed by anyone to audit the election. Verify takes as input the public key $PK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, a tally $\mathbf{X}$, a proof P of correct tallying, and security parameter k. It outputs a bit v, which is 1 if the tally successfully verifies and 0 otherwise. We assume that Verify is deterministic.

Election schemes must satisfy *Correctness*, which asserts that the tally produced by Tally corresponds to the choices input to Vote:

Definition 1 (Correctness)**.** *There exists a negligible function*[8] μ*, such that for all security parameters* k*, integers* n_B *and* n_C*, and choices* $\beta_1, \ldots, \beta_{n_B} \in \{1, \ldots, n_C\}$*, we have if* $\mathbf{Y}$ *is a vector of length* n_C *whose components are all* 0*, then*

```
Pr[$(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
  for $1 \leq i \leq n_B$ do
    $b_i \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta_i, k)$;
    $\mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1$;
  $BB \leftarrow \{b_1, \ldots, b_{n_B}\}$;
  $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k)$ :
  $n_B \leq m_B \wedge n_C \leq m_C \Rightarrow \mathbf{X} = \mathbf{Y}] > 1 - \mu(k)$.
```

Note that Correctness honestly runs the Vote and Tally algorithms, and that it does not involve the adversary. Correctness therefore stipulates that, under ideal conditions, an election scheme does indeed produce the correct tally. Correctness is not actually necessary to achieve verifiability: our definitions of universal verifiability will ensure that, in the presence of the adversary, Verify detects any errors in the tally. But it is reasonable to rule out election schemes that simply do not work properly under ideal conditions.

Election schemes must satisfy *Verify Completeness*, which asserts that tallying produces a tally accepted by the verification algorithm:

Definition 2 (Verify Completeness)**.** *There exists a negligible function* μ*, such that for all security parameters* k*, bulletin*

[3]https://github.com/benadida/helios-server/tree/master/helios_auth/auth_systems, accessed 13 June 2014.

[4]Let $\mathsf{Alg}(in; r)$ denote the output of probabilistic algorithm Alg on input in and random coins r. Let $\mathsf{Alg}(in)$ denote $\mathsf{Alg}(in; r)$, where r is chosen uniformly at random. And let $\leftarrow$ denote assignment.

[5]The maximum ballots and candidate numbers are important to formalize correctness. For instance, Helios requires that the maximum number of ballots is less than or equal to the size of the underlying encryption scheme's message space, and JCJ requires that the maximum number of candidates is less than or equal to the size of the underlying encryption scheme's message space. These maximums may also be useful to algorithms Vote, Tally, and Verify. However, parameters m_B and m_C do not need to be explicitly supplied as input to these algorithms, because they can be included in the public key, where necessary.

[6]Bulletin boards have also been modeled as public broadcast channels [35], [75], [77]. We abstract from the details of channels by employing sets to represent the data sent on them. We favor sets over multisets, because Cortier and Smyth [33], [34] demonstrate attacks against privacy when the bulletin board is modeled as a multiset.

[7]Let $\mathbf{X}[i]$ denote component i of vector $\mathbf{X}$.

[8]Negligible functions, and other standard cryptographic primitives, are defined in the forthcoming technical report.

boards BB, and integers n_C, we have

$$\begin{aligned}&\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k);\\ &\quad (\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k) :\\ &\quad |BB| \leq m_B \wedge n_C \leq m_C \Rightarrow\\ &\qquad \mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1] > 1 - \mu(k).\end{aligned}$$

Verify Completeness stipulates that, under ideal conditions, the tally produced by Tally will actually be accepted by Verify. It turns out that Verify Completeness is necessary to achieve verifiability: without Verify Completeness, election schemes might be vulnerable to biasing attacks, as we show in Section VI-B.

Election schemes must also satisfy *Vote Injectivity*, which asserts that a ballot cannot be interpreted as a vote for more than one candidate:

Definition 3 (Vote Injectivity). *For all security parameters k, public keys $PK_{\mathcal{T}}$, integers n_C, and choices β and β', such that $\beta \neq \beta'$, we have*

$$\begin{aligned}&\Pr[b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k);\\ &\quad b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k) :\\ &\quad b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.\end{aligned}$$

Vote Injectivity ensures that distinct choices are not mapped by Vote to the same ballot. Without Vote Injectivity, an election scheme might produce ballots whose meaning is ambiguous. For example, if $b = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r)$ were defined to be $\beta + r$, then ballot b could be tallied as any well-formed choice β' such that $\beta' = b - r'$ for some r'. But that definition of Vote is prohibited by Vote Injectivity. Vote Injectivity thus helps to ensure that the choices used to construct ballots can be uniquely tallied.

Limitations: Our model of election schemes is sufficient to analyze Helios and (after we extend the model to handle internal authentication in IV-A) JCJ. These are notable schemes, and formally analyzing their verifiability is a novel contribution. But there are other notable schemes that fall outside our model:

- Pret à Voter [24], MarkPledge [72], Scantegrity II [23], and Remotegrity [84] all rely on features implemented with paper, such as scratch-off surfaces and detachable columns.
- Everlasting privacy [70], which requires Vote to output a public ballot and a secret proof, involving temporal information, to the voter.
- Scytl's Pnyx.core ODBP 1.0 [27], which requires the bulletin board to be divided into two parts: a public part visible to all participants, and a secret part visible only to election administrators.

We leave extending our model to these kinds of election schemes as future work.

B. Election verifiability

Election verifiability comprises three aspects: individual, universal, and eligibility verifiability. We express each as an *experiment*, which is an algorithm that outputs 0 or 1. The adversary *wins* an experiment by causing it to output 1.

1) Individual verifiability: In our model of election schemes, all recorded ballots are posted on the bulletin board. So for a voter to verify that her ballot has been recorded, it suffices to enable her to uniquely identify her ballot on the bulletin board.[9]

Individual verifiability experiment $\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)$, where Π denotes an election scheme, $\mathcal{A}$ denotes the adversary, and k denotes a security parameter, therefore challenges $\mathcal{A}$ to generate a scenario in which the voter cannot uniquely identify her ballot:

```
$\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k) =$
1 $(PK_{\mathcal{T}}, n_C, \beta, \beta') \leftarrow \mathcal{A}(k)$;
2 $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$;
3 $b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$;
4 if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$ then
5     return 1
6 else
7     return 0
```

Line 1 asks $\mathcal{A}$ to compute two candidate choices β and β', such that ballots b and b' for those choices, as computed by Vote in lines 2 and 3, are equal. Individual verifiability thus resembles Vote Injectivity (§II-A), but individual verifiability allows choices to be equal and allows $\mathcal{A}$ to choose election parameters.

In essence, Exp-IV-Ext challenges $\mathcal{A}$ to generate a collision from algorithm Vote.[10] If $\mathcal{A}$ cannot win, then voters can uniquely identify their ballots on the bulletin board.

One way to achieve individual verifiability is to base the election scheme on a probabilistic encryption scheme, such as El Gamal [40]. Intuitively, if Vote encrypts the choice using random coins, then it is overwhelmingly unlikely that two votes will result in the same ballot. Our proofs that Helios and JCJ satisfy individual verifiability are based on this idea.

Clash attacks: In a *clash attack* [69], the adversary convinces n voters that a single ballot belongs to them all. The adversary is then free to replace $n-1$ of those ballots on the bulletin board with ballots of his choice.

Some clash attacks are possible because of vulnerabilities in the design of Vote. For example, if Vote simply outputs candidate choice β, then a voter has no way to distinguish her vote for β from another voter's vote for β. Exp-IV-Ext detects clash attacks resulting from vulnerabilities in Vote.

Some clash attacks, however, are possible because the adversary subverts Vote. For example, the adversary might replace some hardware or software, or compromise the random number generator. If any one of these aspects is compromised, then Vote has effectively been changed to a different algo-

[9]Section VIII addresses the complementary issue of whether a recorded ballot corresponds to the candidate choice a voter intended to make.

[10]Exp-IV-Ext can be equivalently formulated as an experiment that challenges $\mathcal{A}$ to predict the output of Vote. See the forthcoming technical report for details.

rithm. Exp-IV-Ext does not detect clash attacks resulting from this kind of compromise.

In short, a voter can verify that her ballot has been recorded if and only if she runs the correct Vote algorithm. We make no guarantees to voters that do not run the correct Vote algorithm. One way to make stronger guarantees is to audit ballots [10], [11]. This would require modeling voting as an interactive protocol with the adversary, rather than as an algorithm. We leave this extension as future work.

2) Universal verifiability: For an election to be universally verifiable, anyone must be able to check that a tally is correct with respect to recorded ballots—that is, the tally represents the choices used to construct the recorded ballots. Because anyone can execute Verify, it suffices that Verify accepts only when that property holds.

Universal verifiability experiment Exp-UV-Ext$(\Pi, \mathcal{A}, k)$ therefore challenges adversary $\mathcal{A}$ to concoct a scenario in which Verify incorrectly accepts:

```
Exp-UV-Ext$(\Pi, \mathcal{A}, k) =$
1 $(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(k)$;
2 $\mathbf{Y} \leftarrow$ correct-tally$(PK_{\mathcal{T}}, BB, n_C, k)$;
3 if $\mathbf{X} \neq \mathbf{Y} \wedge$ Verify$(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$ then
4     return 1
5 else
6     return 0
```

In line 1, $\mathcal{A}$ is challenged to create a bulletin board BB and purported tally $\mathbf{X}$ of that bulletin board. Line 2 constructs the correct tally $\mathbf{Y}$ of BB, and line 3 checks whether Verify accepts an incorrect tally.

Let function *correct-tally* be defined such that for all $PK_{\mathcal{T}}$, BB, n_C, k, ℓ, and $\beta \in \{1, \ldots, n_C\}$,

$$\begin{aligned} &\textit{correct-tally}(PK_{\mathcal{T}}, BB, n_C, k)[\beta] = \ell \\ &\iff \exists^{=\ell} b \in (BB \setminus \{\perp\}) : \\ &\qquad \exists r : b = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r). \end{aligned}$$

That is, component β of vector *correct-tally*$(PK_{\mathcal{T}}, BB, n_C, k)$ equals ℓ iff there exist ℓ ballots on the bulletin board that are votes for candidate β.[11] The vector produced by *correct-tally* must be of length n_C. It follows that the output of *correct-tally* represents the choices used to construct the recorded ballots. Of course, *correct-tally* cannot be computed by a PPT algorithm for typical cryptographic election schemes. But that does not matter, because *correct-tally* is never actually computed as part of an election scheme—its use is solely in the definition of Exp-UV-Ext.

If $\mathcal{A}$ cannot win Exp-UV-Ext, then the tally of ballots on the bulletin board is computed properly. In particular, no ballots could have been omitted from the tally, and at most one candidate choice could have been included in the tally for each ballot. Vote Injectivity ensures that the candidate choice can be uniquely recovered from the ballot.

Exp-UV-Ext uses *correct-tally* instead of Tally, so security analysts must convince themselves that *correct-tally* is indeed correct. Because of the function's simplicity, this should be straightforward. By comparison, Tally algorithms tend to be complicated. For example, compare the complexity of *correct-tally* to Helios's Tally algorithm, which appears in the forthcoming technical report.

By design, Exp-UV-Ext assumes that the ballots on bulletin board BB are exactly the ballots that should be tallied. The external authentication mechanism is assumed to prohibit unauthorized ballots from being posted on BB. Helios makes such an assumption about its external authentication mechanism.

3) Eligibility verifiability: For an election to satisfy eligibility verifiability, anyone must be able to check that every tallied vote was cast by an authorized voter—that is, it must be possible to authenticate ballots. In election schemes with external authentication, a trusted third party authenticates ballots. That third party might convince itself that all tallied ballots have been authenticated, but it cannot convince all other parties. Eligibility verifiability, therefore, is not achievable in election schemes with external authentication.

4) Election verifiability: Putting Exp-IV-Ext and Exp-UV-Ext together, we define election verifiability with external authentication. Let a PPT adversary's *success* Succ(Exp$(\cdot)$) in an experiment Exp$(\cdot)$ be the probability that the adversary wins—that is, Succ(Exp$(\cdot)$) $= \Pr[$Exp$(\cdot) = 1]$.

Definition 4 (Ver-Ext)**.** *An election scheme* Π *satisfies* election verifiability with external authentication (Ver-Ext) *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have* Succ(Exp-IV-Ext$(\Pi, \mathcal{A}, k)$) + Succ(Exp-UV-Ext$(\Pi, \mathcal{A}, k)$) $\leq \mu(k)$.

An election scheme satisfies individual verifiability if Succ(Exp-IV-Ext$(\Pi, \mathcal{A}, k)$) $\leq \mu(k)$, and similarly for universal verifiability.

Example—Toy scheme from nonces: A toy election scheme satisfying Ver-Ext can be based on nonces. Each voter publishes a nonce paired with her choice of candidate to the bulletin board. This scheme illustrates the essence of election verifiability, even though it does not offer any privacy.

Definition 5. *Election scheme* Nonce *is defined as follows:*

- Setup(k) *outputs* $(\perp, \perp, \mathit{poly}(k), \infty)$.[12]
- Vote$(PK_{\mathcal{T}}, n_C, \beta, k)$ *selects a nonce* r *uniformly at random from* $\mathbb{Z}_{2^k}$ *and outputs* (r, β).
- Tally$(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k)$ *computes a vector* $\mathbf{X}$ *of length* n_C*, such that* $\mathbf{X}$ *is a tally of the votes on* BB *for which the nonce is in* $\mathbb{Z}_{2^k}$*, and outputs* $(\mathbf{X}, \perp)$.
- Verify$(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$ *outputs* 1 *if* $(\mathbf{X}, P) =$ Tally$(\perp, \perp, BB, n_C, k)$ *and* 0 *otherwise.*

[11]The definition of *correct-tally* employs a *counting quantifier* [78] denoted $\exists^{=}$. Predicate $(\exists^{=\ell} x : P(x))$ holds exactly when there are ℓ distinct values for x such that $P(x)$ is satisfied. Variable x is bound by the quantifier, whereas ℓ is free.

[12]We write *poly* to denote some polynomial function.

Proposition 1. Nonce *satisfies* Ver-Ext.

Proof sketch. Nonce satisfies individual verifiability, because voters can use their nonce to check that their own ballot appears on the bulletin board. With overwhelming probability, voters will select unique nonces, hence generate distinct ballots. Nonce also satisfies universal verifiability, because plaintext candidate choices are posted on the bulletin board. □

C. Orthogonality

Exp-IV-Ext and Exp-UV-Ext capture orthogonal security properties:

- A scheme that satisfies individual verifiability but violates universal verifiability can be constructed from Nonce by modifying Verify to always output 1. Voters can still check that their own ballot appears. But an adversary can easily win Exp-UV-Ext, because Verify will accept any tally.
- A scheme that satisfies universal verifiability but violates individual verifiability can be constructed from Nonce by removing the nonces, leaving just the voter's choice in the ballots. Call that scheme Choice. Anyone can still verify the tally of the election, but an adversary can easily win Exp-IV-Ext, because two votes for the same candidate will collide.

III. Case Study: Helios

Helios is an open-source, web-based electronic voting system.[13] Helios has been deployed in the real-world: the International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [13], [47], [52], the Catholic University of Louvain used Helios to elect the university president [5], and Princeton University has used Helios to elect several student governments [3], [73].

Attacks have been discovered against the original Helios scheme, and defenses against those attacks have been proposed [16], [17], [33], [34]. For clarity, we write *Helios 2.0* to refer to the Helios scheme as originally proposed [5] and *Helios 4.0* to refer to the version of Helios that incorporates the defenses.[14] When referring in general to both of these schemes, we simply write *Helios*.

To achieve verifiability while maintaining *ballot secrecy* [15], [17], Helios homomorphically encrypts candidate choices. During tallying, all encrypted choices are homomorphically combined[15] into a single ciphertext, which is then decrypted to reveal the tally. Informally, Helios works as follows:

- **Setup.** The tallier generates a key pair for a homomorphic encryption scheme and publishes the public key.[16]
- **Voting.** A voter encrypts her candidate choice with the tallier's public key, and she proves in zero knowledge that the ciphertext contains a well-formed choice. The voter posts her ballot (i.e., ciphertext and proof) on the bulletin board BB. During posting, BB is assumed to correctly authenticate voters.
- **Tallying.** The tallier discards any ballots from the bulletin board for which proofs do not hold. The tallier homomorphically combines the ciphertexts in the remaining ballots, decrypts the homomorphic combination, and proves in zero knowledge that decryption was performed correctly. Finally, the tallier publishes the winning candidate and proof of correct decryption.
- **Verification.** A verifier recomputes the homomorphic combination and checks all the zero-knowledge proofs.

We give a formal description of Helios 4.0 in the forthcoming technical report.[17] Using that formalization, we can prove that Helios 4.0 is verifiable:

Theorem 1. *Helios 4.0 satisfies* Ver-Ext.

Proof sketch. Helios 4.0 satisfies individual verifiability, because the probabilistic encryption scheme ensures that ballots are unique, with overwhelming probability. And Helios 4.0 satisfies universal verifiability, because the zero-knowledge proofs can be publicly verified. □

A formal proof of Theorem 1 appears in the forthcoming technical report. The proof assumes the random oracle model [8].

We would not expect Ver-Ext to hold for earlier revisions of Helios, because they are known to be vulnerable to attacks against verifiability [17]. Accordingly, we prove that Helios 2.0 does not satisfy Ver-Ext in the forthcoming technical report.

IV. Internal Authentication

Some election schemes implement their own authentication mechanisms. JCJ [55]–[57] and Civitas [28], for example, authenticate ballots based on *credentials* issued to voters by a registration authority. Schemes with this kind of internal authentication enable verification of whether tallied ballots were cast by authorized voters.

A. Election scheme

A *registrar* is responsible for issuing authentication *credentials* to voters. Each voter is associated with a credential pair (pk, sk). The voter uses private credential sk to construct a ballot. Public credential pk is used during tallying and verification. Let L denote the *electoral roll*, which is the set of all public credentials.

An *election scheme with internal authentication*, which henceforth in this section we abbreviate as "election scheme,"

[13]https://vote.heliosvoting.org/

[14]Our formalization of Helios 4.0 is based on the specification [4] for the next release. This specification incorporates proposals by Cortier and Smyth [34] for non-malleable ballots and by Bernhard et al. [17] to replace the weak Fiat–Shamir transformation with the strong Fiat–Shamir transformation.

[15]The homomorphic combination of ciphertexts is straightforward for two-candidate elections [9], [14], [29], [49], [76], since choices (e.g., "yes" or "no") can be encoded as 1 or 0. Multi-candidate elections are also possible [14], [36], [48].

[16]Helios permits the tallier's role to be distributed amongst several talliers. For simplicity, we consider only a single tallier in this paper.

[17]Our formalization is the first cryptographic description of Helios 4.0, hence an additional contribution of this work.

is a tuple (Setup, Register, Vote, Tally, Verify) of PPT algorithms. The algorithms are now denoted as follows:

- $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$
- $(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$
- $b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$
- $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k)$
- $v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k)$

Setup is unchanged from election schemes with external authentication (cf. §II-A). The only change to Vote is that it now accepts private credential sk as input. Similarly, the only change to Tally and Verify is that they now accept electoral roll L as input.

Register is executed by the registrar. It takes as input the public key $PK_{\mathcal{T}}$ of the tallier and security parameter k. It outputs a credential pair (pk, sk). After all voters have been registered, the registrar certifies the electoral roll, perhaps by digitally signing and publishing it.[18]

Election schemes must continue to satisfy Correctness, Verify Completeness, and Vote Injectivity, which we update to include private credentials and the electoral roll:

Definition 6 (Correctness)**.** *There exists a negligible function μ, such that for all security parameters k, integers n_B and n_C, and choices $\beta_1, \ldots, \beta_{n_B} \in \{1, \ldots, n_C\}$, we have if $\mathbf{Y}$ is a vector of length n_C whose components are all 0, then*

```
Pr[$(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
    for $1 \le i \le n_B$ do
        $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
        $b_i \leftarrow \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k)$;
        $\mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1$;
    $L \leftarrow \{pk_1, \ldots, pk_{n_B}\}$;
    $BB \leftarrow \{b_1, \ldots, b_{n_B}\}$;
    $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k)$ :
    $n_B \le m_B \wedge n_C \le m_C \Rightarrow \mathbf{X} = \mathbf{Y}] > 1 - \mu(k)$.
```

Definition 7 (Verify Completeness)**.** *There exists a negligible function μ, such that for all security parameters k, bulletin boards BB, and integers n_C and n_V, we have*

```
Pr[$(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
    for $1 \le i \le n_V$ do $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
    $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
    $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k)$ :
    $|BB| \le m_B \wedge n_C \le m_C \Rightarrow$
        $\mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k) = 1] > 1 - \mu(k)$.
```

Definition 8 (Vote Injectivity)**.** *For all security parameters k, public keys $PK_{\mathcal{T}}$, integers n_C and choices β and β', such that $\beta \neq \beta'$, we have*

$$
\begin{aligned}
\Pr[&(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k);\\
&(pk', sk') \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k);\\
&b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k);\\
&b' \leftarrow \mathsf{Vote}(sk', PK_{\mathcal{T}}, n_C, \beta', k) :\\
&b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.
\end{aligned}
$$

B. Election verifiability

Recall (from §II-B) that election verifiability is expressed with experiments, and that an adversary wins by causing an experiment to output 1. We henceforth assume that the adversary is *stateful*—that is, information persists across invocations of the adversary in a single experiment. Our experiments in Section II did not need this assumption, because they never invoked the adversary more than once.

1) Individual verifiability: The individual verifiability experiment again challenges adversary $\mathcal{A}$ to generate a scenario in which the voter could not uniquely identify her ballot:[19]

```
Exp-IV-Int($\Pi, \mathcal{A}, k$) =
1  $(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k)$;
2  for $1 \le i \le n_V$ do $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$
3  $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
4  $Crpt \leftarrow \emptyset$;
5  $(n_C, \beta, \beta', i, j) \leftarrow \mathcal{A}^C(L)$;
6  $b \leftarrow \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k)$;
7  $b' \leftarrow \mathsf{Vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta', k)$;
8  if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$
   $\wedge\, i \neq j \wedge sk_i \notin Crpt \wedge sk_j \notin Crpt$ then
9      return 1
10 else
11     return 0
```

The main differences from the corresponding experiment for external authentication (§II-B1) are that voters are registered in line 2, and that $\mathcal{A}$ is given access to an oracle C in line 5. That oracle is used to model $\mathcal{A}$ corrupting voters and learning their private credentials. On invocation $C(\ell)$, where $1 \le \ell \le n_V$, the oracle records that voter ℓ is corrupted by updating $Crpt$ to be $Crpt \cup \{sk_\ell\}$ and outputs sk_ℓ.

In line 5, $\mathcal{A}$ must output two candidate choices and two voter indices, such that ballots computed for those are equal. Those indices must be legal with respect to n_V, but we elide that detail from the experiment for simplicity. In line 8, $\mathcal{A}$ wins only if the voter indices $\mathcal{A}$ output were not corrupted during the experiment, meaning that those voters never revealed their private credentials to $\mathcal{A}$.

2) Universal verifiability: The universal verifiability experiment again challenges $\mathcal{A}$ to concoct a scenario in which Verify incorrectly accepts a tally:

[18]It might at first seem surprising that Register does not require the registrar to provide any private keys as input. But in constructions of election schemes with internal authentication, e.g., [28], [57], the registrar does not sign credential pairs with its own private key. Rather, the registrar signs the electoral roll.

[19]Unlike Exp-IV-Ext, a variant of Exp-IV-Int that challenges $\mathcal{A}$ to predict the output of Vote is strictly stronger. See the forthcoming technical report for details.

```
Exp-UV-Int$(\Pi, \mathcal{A}, k)$ =
1  $(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k)$;
2  for $1 \leq i \leq n_V$ do $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$
3  $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
4  $M \leftarrow \{(pk_1, sk_1), \ldots, (pk_{n_V}, sk_{n_V})\}$;
5  $(BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(M)$;
6  $\mathbf{Y} \leftarrow \mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, M, n_C, k)$;
7  if $\mathbf{X} \neq \mathbf{Y} \wedge \mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k) = 1$ then
8      return 1
9  else
10     return 0
```

The main differences from the corresponding experiment for external authentication (§II-B2) are that voters are registered in line 2, and their credential pairs are used in the rest of the experiment.

Function *correct-tally* is now modified to tally only authorized ballots. A ballot is *authorized* if it is constructed with a private credential from M, and that private credential was not used to construct any other ballot on BB. By comparison, the original *correct-tally* function (§II-B2) tallies all the ballots on BB.

Formally, let function *correct-tally* now be defined such that for all $PK_{\mathcal{T}}$, BB, M, n_C, k, ℓ, and $\beta \in \{1, \ldots, n_C\}$,

$$\begin{aligned} &\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, M, n_C, k)[\beta] = \ell \\ &\iff \exists^{=\ell} b \in \mathit{authorized}(PK_{\mathcal{T}}, (BB \setminus \{\perp\}), M, n_C, k) : \\ &\qquad \exists sk, r : b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r). \end{aligned}$$

And let *authorized* be defined as follows:

$$\begin{aligned} &\mathit{authorized}(PK_{\mathcal{T}}, BB, M, n_C, k) = \\ &\quad \{b : b \in BB \\ &\quad\quad \wedge \exists pk, sk, \beta, r : b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r) \\ &\quad\quad\quad \wedge (pk, sk) \in M \wedge \neg \exists b', \beta', r' : b' \in (BB \setminus \{b\}) \\ &\quad\quad\quad\quad \wedge b' = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta', k; r')\}. \end{aligned}$$

Function *authorized* discards all revotes—that is, if there is more than one ballot submitted with a private credential sk, then all ballots submitted under that credential are discarded. Therefore, election schemes that permit revoting cannot by analyzed with this definition of *authorized*. But alternative definitions of *authorized* are possible—for example, if ballots were timestamped, *authorized* could discard all but the most recent ballot submitted under a particular credential.

3) Eligibility verifiability: Recall (from §II-B3) that for an election scheme to satisfy eligibility verifiability, anyone must be able to check that every tallied vote was cast by an authorized voter—that is, it must be possible to authenticate ballots. Because voters are issued credential pairs that can be used to authenticate ballots, it suffices to ensure that knowledge of a private credential is necessary to construct an authentic ballot.

The eligibility verifiability experiment therefore challenges $\mathcal{A}$ to produce a ballot under a private credential that $\mathcal{A}$ does not know:

```
Exp-EV-Int$(\Pi, \mathcal{A}, k)$ =
1  $(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k)$;
2  for $1 \leq i \leq n_V$ do $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
3  $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
4  $\mathit{Crpt} \leftarrow \emptyset$; $\mathit{Rvld} \leftarrow \emptyset$;
5  $(n_C, \beta, i, b) \leftarrow \mathcal{A}^{C,R}(L)$;
6  if $\exists r : b = \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp$
     $\wedge\, b \notin \mathit{Rvld} \wedge sk_i \notin \mathit{Crpt}$ then
7      return 1
8  else
9      return 0
```

In line 1, $\mathcal{A}$ chooses the tallier's public key and the number of voters. Line 2 registers voters. $\mathcal{A}$ is not permitted to influence registration while it is in progress.[20] In particular, $\mathcal{A}$ is not permitted to choose credential pairs, because by doing so $\mathcal{A}$ could trivially win the experiment.

Line 4 initializes two sets: *Crpt* is a set of voters who have been corrupted, meaning that $\mathcal{A}$ has learned their private credential, and *Rvld* is a set of ballots that have been revealed to $\mathcal{A}$. The former set is useful to track, because $\mathcal{A}$ might coerce some voters into revealing their private credentials. The latter set is useful to track, because $\mathcal{A}$ will naturally learn some ballots by observing them on the bulletin board.

Line 5 challenges $\mathcal{A}$ to produce a ballot b with the help of two oracles. Oracle C is the same oracle as in Exp-IV-Int (cf. §IV-B1); it leaks the private credentials of corrupted voters to $\mathcal{A}$. Oracle R reveals ballots. On invocation $R(i, \beta, n_C)$, where $1 \leq i \leq n_V$, oracle R does the following:

1) Computes a ballot b that represents a vote for candidate β by a voter with private credential sk_i, that is, $b \leftarrow \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k)$.
2) Records b as being revealed by updating *Rvld* to be $\mathit{Rvld} \cup \{b\}$.
3) Outputs b.

In line 6, $\mathcal{A}$ wins if the ballot it produced is the output of Vote for a voter that $\mathcal{A}$ did not corrupt, and if that ballot was not revealed. If $\mathcal{A}$ cannot succeed in this experiment, then knowledge of a private credential is necessary to construct a ballot, hence only authorized votes are tallied.

4) Election verifiability: With Exp-IV-Int, Exp-UV-Int, and Exp-EV-Int, we define election verifiability with internal authentication.

Definition 9 (Ver-Int)**.** *An election scheme* Π *satisfies* election verifiability with internal authentication (Ver-Int) *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have*

$$\begin{aligned} &\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, k)) \\ &+ \mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Int}(\Pi, \mathcal{A}, k)) \\ &+ \mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k)) \leq \mu(k). \end{aligned}$$

[20]Küsters and Truderung [65] explore some consequences of permitting adversarial influence during registration.

An election scheme satisfies eligibility verifiability if $\mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Ext}(\Pi, \mathcal{A}, k)) \leq \mu(k)$, and similarly for individual and universal verifiability.

Example—Toy scheme from digital signatures: A toy election scheme satisfying Ver-Int can be based on a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. Each voter publishes her signed candidate choice on the bulletin board.

Definition 10. *Election scheme* Sig *is defined as follows:*

- $\mathsf{Setup}(k)$ *outputs* $(\perp, \perp, poly(k), \infty)$.
- $\mathsf{Register}(PK_\mathcal{T}, k)$ *computes* $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ *and outputs* (pk, sk).
- $\mathsf{Vote}(sk, PK_\mathcal{T}, n_C, \beta, k)$ *outputs* $(\beta, \mathsf{Sign}(sk, \beta))$.
- $\mathsf{Tally}(PK_\mathcal{T}, SK_\mathcal{T}, BB, L, n_C, k)$ *computes a vector* $\mathbf{X}$ *of length* n_C*, such that* $\mathbf{X}$ *is a tally of all the ballots on* BB *that are signed by distinct private keys whose corresponding public keys appear in* L*, and outputs* $(\mathbf{X}, \perp)$.
- $\mathsf{Verify}(PK_\mathcal{T}, BB, L, n_C, \mathbf{X}, P, k)$ *outputs* 1 *if* $(\mathbf{X}, P) = \mathsf{Tally}(\perp, \perp, BB, L, n_C, \perp)$ *and* 0 *otherwise.*

The verifiability of Sig follows from the security of the underlying signature scheme.

Proposition 2. *If* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is a signature scheme satisfying existential unforgeablility under adaptive chosen-message attack, then* Sig *satisfies* Ver-Int.

Proof sketch. Sig satisfies individual verifiability, because voters can verify that their signed choices appear on the bulletin board. Sig satisfies universal verifiability, because signed plaintext choices are posted on BB. Finally, Sig satisfies eligibility verifiability, because anyone can check that the signed choices belong to registered voters. □

Example—Helios-C: Helios-C [32] is a variant of Helios for two candidate elections in which ballots are digitally signed.[21] Informally, Helios-C works as follows:

- **Setup.** As in Section III.
- **Registration.** To register a voter, the registrar generates a key pair for a signature scheme and sends the private key to the voter. After all voters are registered, the registrar publishes electoral roll L.
- **Voting.** A voter generates a ciphertext and proof as in Section III, signs the ciphertext and proof with her private key, and posts the ciphertext, proof, and signature on the bulletin board.
- **Tallying.** The tallier discards all ballots on the bulletin board that are not signed by distinct private keys whose corresponding public keys appear in L. The remaining ciphertexts and proofs are processed as in Section III.
- **Verification.** A verifier first discards ballots that are not properly signed, then continues as in Section III.

Helios-C satisfies Ver-Int: individual and universal verifiability follow from Theorem 1, and eligibility verifiability is satisfied because anyone can check that the signed choices belong to registered voters, as in Proposition 2. We omit a formal proof.

[21] https://github.com/glondu/helios-server

Line	IV	UV	EV	Scheme
1	✗	✗	✗	AlwaysVerify(IgnoreCreds(Choice))
2	✗	✗	✓	—
3	✗	✓	✗	IgnoreCreds(Choice)
4	✗	✓	✓	—
5	✓	✗	✗	AlwaysVerify(IgnoreCreds(Nonce))
6	✓	✗	✓	AlwaysVerify(Sig)
7	✓	✓	✗	Malleable Sig
8	✓	✓	✓	Sig

TABLE I
ELECTION SCHEMES THAT SATISFY EACH COMBINATION OF INDIVIDUAL, UNIVERSAL AND ELIGIBILITY VERIFIABILITY

C. Orthogonality

If an election scheme satisfies eligibility verifiability, then no one can construct a ballot that appears to be associated with public credential pk unless they know private credential sk. That means that a voter can uniquely identify her ballot, because no one else knows her private credential. Eligibility verifiability therefore implies individual verifiability.

Theorem 2. *If an election scheme* Π *satisfies eligibility verifiability, then* Π *also satisfies individual verifiability.*

The proof of Theorem 2 appears in the forthcoming technical report.

Otherwise, Exp-IV-Int, Exp-UV-Int, and Exp-EV-Int capture orthogonal security properties, as shown in Table I. In that table, $\mathsf{AlwaysVerify}(\cdot)$ is a function that transforms an election scheme by compromising Verify to always return 1. Thus, $\mathsf{AlwaysVerify}(\Pi)$ is guaranteed not to satisfy Exp-UV-Int. Similarly, $\mathsf{IgnoreCreds}(\cdot)$ is a function that accepts as input an election scheme with external authentication and returns as output an election scheme with internal authentication. The resulting scheme, however, simply ignores credentials altogether: Register returns $(\perp, \perp)$, Vote ignores sk, and Tally and Verify ignore L. Thus, $\mathsf{IgnoreCreds}(\Pi)$ is guaranteed not to satisfy Exp-EV-Int. Using those functions, we briefly explain each line of the table:

1) Recall (from §II-C) that Choice is the election scheme in which ballots contain only the plaintext candidate choice. That scheme satisfies Exp-UV-Ext but not Exp-IV-Ext. By compromising Verify, we obtain a scheme that satisfies no properties.
2) By Theorem 2, this situation is impossible.
3) Compared to line 1 of Table I, this scheme also satisfies Exp-UV-Int, because Verify is not compromised.
4) By Theorem 2, this situation is impossible.
5) Recall (from §II-B4) that Nonce is the election scheme in which ballots contain a nonce and a plaintext candidate choice. That scheme satisfies Exp-IV-Ext and Exp-UV-Ext. Moreover, IgnoreCreds(Nonce) satisfies Exp-IV-Int and Exp-UV-Int. By compromising Verify, we obtain a scheme that satisfies only Exp-IV-Int.

6) Sig satisfies all three properties. By compromising Verify, we obtain a scheme that satisfies only Exp-IV-Int and Exp-EV-Int.
7) By making Sig's underlying signature scheme malleable,[22] we could obtain a scheme that does not satisfy Exp-EV-Int, because the adversary could construct a valid ballot out of a revealed ballot. But the scheme would continue to satisfy Exp-IV-Int and Exp-UV-Int.
8) Sig satisfies all three properties.

V. Case Study: JCJ and Civitas

JCJ [55]–[57] (named for its designers, Juels, Catalano, and Jakobsson) is a *coercion-resistant* election scheme, meaning voters cannot prove whether or how they voted, even if they can interact with the adversary while voting. Coercion resistance protects elections from improper influence by adversaries.

To achieve verifiability and coercion resistance, JCJ uses verifiable *mixnets*, which anonymize a set of messages.[23] During tallying, all encrypted choices are anonymized by a mixnet, then all choices are decrypted. The tally is computed from the decrypted choices. Informally, JCJ works as follows:

- **Setup.** The tallier generates a key pair $(PK_\mathcal{T}, SK_\mathcal{T})$ for an encryption scheme and publishes the public key.
- **Registration.** To register a voter, the registrar generates a nonce, which is sent to the voter and serves as the private credential.[24] The public credential is computed as an encryption of the private credential with $PK_\mathcal{T}$. After all voters are registered, the registrar publishes the electoral roll.
- **Voting.** A voter encrypts her candidate choice with $PK_\mathcal{T}$. She also encrypts her private credential with $PK_\mathcal{T}$. She proves in zero-knowledge that she simultaneously knows both plaintexts, and that her choice is well-formed. The voter posts her ballot (i.e., both ciphertexts and the proof) on the bulletin board.
- **Tallying.** The tallier discards any ballots from the bulletin board for which the zero-knowledge proofs do not verify. All unauthorized ballots are then discarded through a combination of protocols that includes verifiable mixnets and *plaintext equivalence tests* (PETs) [53]. PETs enable proof that two ciphertexts contain the same plaintext without revealing that plaintext. The tallier decrypts and publishes the remaining ballots, along with a proof that decryption was performed correctly.
- **Verification.** A verifier checks all the proofs included in ballots, and all the proofs published during tallying.

The forthcoming technical report gives a formal description of JCJ. That formalization satisfies individual and universal verifiability, assuming that the cryptographic primitives satisfy certain properties that we identify. But the formalization fails to satisfy eligibility verifiability, because knowledge of the tallier's private key $SK_\mathcal{T}$ suffices to construct ballots that appear authentic: with $SK_\mathcal{T}$, any public credential can be decrypted to discover the corresponding private credential. Note that Exp-EV-Int permits an adversary $\mathcal{A}$ to choose the tallier's key pair, so $\mathcal{A}$ does know $SK_\mathcal{T}$ hence can construct a ballot that suffices to win Exp-EV-Int.

We can nonetheless prove that JCJ satisfies a weaker variant of eligibility verifiability. Consider the following experiment, which does not permit the adversary to choose the tallier's key pair:

```
Exp-EV-Int-Weak$(\Pi, \mathcal{A}, k)$ =
1  $(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
2  $n_V \leftarrow \mathcal{A}(PK_\mathcal{T}, k)$;
3  for $1 \le i \le n_V$ do $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k)$
4  $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
5  $Crpt \leftarrow \emptyset$; $Rvld \leftarrow \emptyset$;
6  $(n_C, \beta, i, b) \leftarrow \mathcal{A}^{C,R}(L)$;
7  if $\exists r : b = \mathsf{Vote}(sk_i, PK_\mathcal{T}, n_C, \beta, k; r) \wedge b \neq \perp$
   $\wedge\, b \notin Rvld \wedge sk_i \notin Crpt$ then
8      return 1
9  else
10     return 0
```

Line 1 of Exp-EV-Int has been refactored into lines 1 and 2 of Exp-EV-Int-Weak. In line 1 of Exp-EV-Int-Weak, keys are generated by the experiment. In line 2, $\mathcal{A}$ is given the public key but not the private key.

Using Exp-EV-Int-Weak, we define a weaker variant of Ver-Int and prove that JCJ satisfies it:

Definition 11 (Ver-Int-Weak)**.** *An election scheme* Π *satisfies* weak election verifiability with internal authentication (Ver-Int-Weak) *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have*

$$\begin{aligned} &\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, k)) \\ &+ \mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Int}(\Pi, \mathcal{A}, k)) \\ &+ \mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int\text{-}Weak}(\Pi, \mathcal{A}, k)) \le \mu(k). \end{aligned}$$

Theorem 3. *JCJ satisfies* Ver-Int-Weak.

Proof sketch. JCJ satisfies individual verifiability, because the probabilistic encryption scheme ensures that ballots are unique, with overwhelming probability. JCJ satisfies universal verifiability, because the proofs produced throughout tallying can be publicly verified. And JCJ satisfies eligibility verifiability, because $\mathcal{A}$ cannot construct new ballots without knowing a voter's private credential or the tallier's private key. □

A formal proof of Theorem 3 appears in the forthcoming technical report. The proof assumes the random oracle model [8].

The Civitas scheme refines the JCJ scheme. The refinements relevant to election verifiability are (i) an implementation

[22]Given a message m and signature σ, a *malleable* signature scheme permits computation of a signature σ' on a related message m' [20]. The malleable signature scheme Sig used in line 7 of Table I would need to enable an adversary to transform a signature on a well-formed candidate β into a signature on a distinct, well-formed candidate β'.

[23]Chaum [21] introduced mixnets. Adida [1] surveys verifiable mixnets.

[24]JCJ permits the registrar's role to be distributed among several registrars. For simplicity, we consider only a single registrar in this paper.

of a distributed registration protocol, (ii) a mixnet based on randomized partial checking (RPC), and (iii) usage of some different zero-knowledge proofs than JCJ. We leave a proof that the full Civitas scheme satisfies Ver-Int as future work.[25]

VI. New classes of attack

Our definitions of election verifiability improve upon existing definitions by detecting two previously unidentified classes of attack:

- *Collusion attacks.* An election scheme's tallying and verification procedures might collude to accept incorrect tallies.
- *Biasing attacks.* An election scheme's verification procedure might reject some legitimate tallies.

Although an honestly-designed election scheme would hopefully not exhibit these vulnerabilities, it is the job of verifiability definitions to detect malicious schemes, regardless of whether vulnerabilities are due to malice or slips. So definitions of election verifiability should preclude collusion and biasing attacks.

A. Collusion Attacks

Here are two examples of potential collusion attacks:

- **Ballot stuffing.** Tally behaves normally, but adds κ votes for candidate β. Verify subtracts κ votes from β, then proceeds with verification as normal. Elections thus verify as normal, except that candidate β receives extra votes.
- **Backdoor tally replacement.** Tally and Verify behave normally, unless a *backdoor* value is posted on the bulletin board BB. For example, if $(SK_{\mathcal{T}}, \mathbf{X}^*)$ appears on BB, then Tally and Verify both ignore the correct tally and instead replace it with tally $\mathbf{X}^*$. Value $SK_{\mathcal{T}}$ is the backdoor here; it cannot appear on BB (except with negligible probability) unless the tallier is malicious.

Ballot stuffing is detected by our definitions of correctness (§II-A and §IV-A), because these definitions require that the tally produced by Tally corresponds to the choices encapsulated in ballots on the bulletin board. Backdoor tally replacement is detected by our definitions of universal verifiability (§II-B2 and §IV-B2), because those definitions require Verify to accept only those tallies that correspond to a correct tally of the bulletin board.

Prior definitions of election verifiability [32], [57] do not rule out collusion attacks. We show, next, that the definition of election verifiability by Juels et al. [57] fails to detect ballot stuffing and backdoor tally replacement, and that the definition by Cortier et al. [32] fails to detect backdoor tally replacement.

Juels et al. [57] formalize definitions that we name *JCJ-correctness* and *JCJ-verifiability*. We restate those definitions in the forthcoming technical report. JCJ-correctness is meant to capture that "$\mathcal{A}$ cannot pre-empt, alter, or cancel the votes of honest voters [and] that $\mathcal{A}$ cannot cause voters to cast ballots resulting in double voting" [57, p. 45]. JCJ-verifiability is "the ability for any player to check whether the tally...has been correctly computed" [57, p. 46].

To show that the JCJ definitions fail to detect collusion attacks, we first formalize the ballot stuffing attack. An election scheme $\Pi = (\ldots, \mathsf{Tally}, \mathsf{Verify})$ can be modified to derive a ballot-stuffing election scheme $\mathsf{Stuff}(\Pi, \beta, \kappa) = (\ldots, \mathsf{Tally}_S, \mathsf{Verify}_S)$, which adds κ votes to β, as follows:[26]

- $\mathsf{Tally}_S(\ldots)$ computes $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(\ldots)$ and outputs $(\mathtt{Add}(\mathbf{X}, \beta, \kappa), P)$, and
- $\mathsf{Verify}_S(\ldots, \mathbf{X}, P, \ldots)$ computes $v \leftarrow \mathsf{Verify}(\ldots, \mathtt{Add}(\mathbf{X}, \beta, -\kappa), P, \ldots)$ and outputs v,

where $\mathtt{Add}(\mathbf{X}, \beta, \kappa) = (\mathbf{X}[1], \ldots, \mathbf{X}[\beta - 1], \mathbf{X}[\beta] + \kappa, \mathbf{X}[\beta + 1], \ldots, \mathbf{X}[|\mathbf{X}|])$.[27]

Proposition 3. *Given an election scheme Π satisfying JCJ-correctness and JCJ-verifiability, and integers β and κ, it holds that $\mathsf{Stuff}(\Pi, \beta, \kappa)$ satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in the forthcoming technical report.

Second, we formalize the backdoor tally replacement attack. An election scheme $\Pi = (\ldots, \mathsf{Tally}, \mathsf{Verify})$ can be modified to derive an election scheme $\mathsf{Backdoor}(\Pi) = (\ldots, \mathsf{Tally}_B, \mathsf{Verify}_B)$, as follows:

- $\mathsf{Tally}_B(\ldots, SK_{\mathcal{T}}, BB, \ldots)$ checks whether there exists a unique $\mathbf{X}^*$ such that $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$. If so, Tally_B outputs $(\mathbf{X}^*, \perp)$. Otherwise, Tally_B computes
$$(\mathbf{X}, P) \leftarrow \mathsf{Tally}(\ldots, SK_{\mathcal{T}}, BB, \ldots)$$
and outputs $(\mathbf{X}, P)$.
- $\mathsf{Verify}_B(PK_{\mathcal{T}}, \ldots, BB, \ldots, \mathbf{X}, P, \ldots)$ checks whether there exists a unique $\mathbf{X}^*$ such that $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$.[28] If so, Verify_B outputs 1. Otherwise, Verify_B computes
$$v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, \ldots, BB, \ldots, \mathbf{X}, P, \ldots)$$
and outputs v.

Proposition 4. *Given an election scheme Π satisfying JCJ-correctness and JCJ-verifiability that does not leak the tallier's private key, it holds that $\mathsf{Backdoor}(\Pi)$ satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in the forthcoming technical report, where we also formally define key leakage.

Cortier et al. [32] propose definitions similar to *JCJ-verifiability* and insist that election schemes must satisfy their notions of correctness and partial tallying. Ballot stuffing is detected by their correctness property, but backdoor tally replacement is not. The ideas remain the same, so we omit formalized results. We have reported these findings to the original authors [30], [43], [44].

[25] In that proof, it would be necessary to assume the RPC construction satisfies the definition of mixnets given in the forthcoming technical report. Work by Khazaei and Wikström [58] suggests that actually proving satisfaction is unlikely to be possible. Alternatively, the mixnet could be replaced by one based on zero-knowledge proofs [42], [71].

[26] We omit many of the parameters of Tally and Verify here for simplicity; see the forthcoming technical report for details.

[27] Let $|\mathbf{X}|$ denote the length of vector $\mathbf{X}$.

[28] Verify_B also needs to check that $SK_{\mathcal{T}}$ is the private key corresponding to $PK_{\mathcal{T}}$. We omit formalizing this detail, but note that it is straightforward for real-world encryption schemes such as El Gamal and RSA.

B. Biasing attacks

Here are three formalizations of biasing attacks, derived from an election scheme $\Pi = (\dots, \mathsf{Verify})$.

- **Reject All.** Let $\mathsf{Reject}(\Pi)$ be $(\dots, \mathsf{Verify}_R)$, where Verify_R always outputs 0. Verify_R therefore always rejects, hence no election can ever be considered valid.
- **Selective Reject.** Let ε be a distinguished value. Let $\mathsf{Selective}(\Pi, \varepsilon)$ be $(\dots, \mathsf{Verify}_R)$, where $\mathsf{Verify}_R(\dots, BB, \dots)$ computes
$$v \leftarrow \mathsf{Verify}(\dots, BB, \dots)$$
and outputs 1 if both $v = 1$ and $\varepsilon \notin BB$. Otherwise, Verify_R outputs 0. Verify_R therefore rejects if ε appears on the bulletin board, hence some elections can be invalidated.
- **Biased Reject.** Suppose Z is a set of tallies. Let $\mathsf{Bias}(\Pi, Z)$ be $(\dots, \mathsf{Verify}_R)$, where $\mathsf{Verify}_R(\dots, \mathbf{X}, \dots)$ computes
$$v \leftarrow \mathsf{Verify}(\dots, \mathbf{X}, \dots)$$
and outputs 1 if both $v = 1$ and $\mathbf{X} \in Z$. Otherwise, Verify_R outputs 0. Verify_R therefore only accepts a subset of the tallies accepted by Verify, hence biases tallies toward Z.

These formalizations do not satisfy our definitions of Verify Completeness (§II-A and §IV-A), hence, our definitions of verifiability detect these biasing attacks.

The definition of verifiability by Juels et al. [57] fails to detect all three of the above attacks, because that definition has no notion of Verify Completeness. For example, it is vulnerable to Biased Reject attacks:

Proposition 5. *Given an election scheme* Π *satisfying JCJ-correctness and JCJ-verifiability, and given a multiset* Z*, it holds that* $\mathsf{Bias}(\Pi, Z)$ *satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in the forthcoming technical report.

The definition of verifiability by Cortier et al. [32] detects Biased Reject and Reject All attacks, but fails to detect Selective Reject attacks, because that definition's notion of Verify Completeness does not quantify over all bulletin boards. Again, the ideas remain the same, so we omit formalized results. We have reported these findings to the original authors [30], [43], [44].[29]

VII. Related Work

Kiayias [59] presents an overview of security properties for election schemes. Many election schemes in the literature state properties called correctness, accuracy, or (universal) verifiability without formally defining those terms.

In the computational model, Juels et al. [55]–[57] and Cortier et al. [32] give game-based definitions of verifiability. As we have shown, those definitions fail to detect biasing and collusion attacks (cf. §VI). Definitions of universal verifiability (which is just one aspect of election verifiability) in the computational model seem to originate with Benaloh and Tuinstra [12], who define a *correctness* property that says every participant is convinced that the tally is accurate with respect to the votes cast, and with Cohen and Fischer [29], who define *verifiability* to mean that there exists a *check* function that returns good iff the announced tally of the election corresponds to the cast votes.

Also in the computational model, Groth [46], and Moran and Naor [70], state definitions of verifiability in terms of *universal composability* [19]. These definitions involve defining an *ideal functionality*; part of that is similar to our *correct-tally* function. Groth's definition does not guarantee universal verifiability [46, p. 2], but Moran and Naor's does [70, p. 386].

In the symbolic model, Smyth et al. [82] define the first definition of election verifiability. This definition is amenable to automated reasoning, but is stronger than necessary and cannot be satisfied by many election schemes, including Helios and Civitas. Kremer et al. [61] overcome this limitation with a weaker definition that sacrifices amenability to automated reasoning, and Smyth [79, §3] extends this definition. Dreier et al. have adapted election verifiability to auction [39] and examination [38] systems.

Also in the symbolic model, Kremer and Ryan [60] and Backes et al. [7] formalize definitions of *eligibility*. For both definitions, if a voting protocol satisfies the definition, then only ballots cast by authorized voters will be tallied. These definitions are not intended to provide assurances if the election authorities are dishonest. For example, the definition of Kremer and Ryan does not detect whether corrupt election authorities insert votes [60, §5.2]. Likewise, the definition of Backes et al. assumes that election authorities are honest [7, §3].

Our definition of election verifiability follows Smyth et al. [61], [79], [82] by deconstructing it into individual, universal, and eligibility verifiability. Other deconstructions of election verifiability are possible. For example, Adida and Neff [6, §2] identify four aspects of verifiability:

- *Cast as intended:* the ballot is cast at the polling station as the voter intended.
- *Recorded as cast:* cast ballots are preserved with integrity through the ballot collection process.
- *Counted as recorded:* recorded ballots are counted correctly.
- *Eligible voter verification:* only eligible voters can cast a ballot in the first place.

Those definitions are not mathematical, so we cannot attempt a precise comparison. Nonetheless, eligibility verifiability and eligible voter verification seem to be addressing similar concerns. Likewise, individual and universal verifiability together seem to be addressing concerns similar to that of recorded as cast and counted as recorded together. Recorded as cast, in our work, reduces to the bulletin board preserving ballots with integrity—a property that we have assumed, because

[29]We reported problems with the peer-reviewed definitions [32] in [44] and problems with an earlier definition [31] in [30], [43].

cryptographic election schemes assume it, too. Peters [75] and Sandler and Wallach [77] propose ways to construct secure bulletin boards. We postpone a discussion of cast as intended to Section VIII.

Privacy properties [37], [57], [67], [68], [80], [81]—such as ballot secrecy, receipt freeness, and coercion resistance—complement verifiability. Chevallier-Mames et al. [25], [26] and Hosp and Vora [50], [51] show an incompatibility result: election schemes cannot unconditionally satisfy privacy and universal verifiability. But weaker versions of these properties can hold simultaneously, as can be witnessed from Theorems 1 and 3 coupled with existing privacy results such as the ballot secrecy proofs for Helios 4.0 [17, Theorem 3], [15, Theorem 6.12], and the coercion resistance proof for JCJ [57, §5].

Comparison with global verifiability: Küsters et al. [66], [67], [69] present a definition of *global verifiability* that can be used with any kind of protocol, not just electronic voting protocols. To analyze the verifiability of a protocol, users of this definition must themselves formalize *goals*, which are properties required to hold in every run of the protocol. For example, a goal γ_ℓ is presented in a case study [67, §5.2] of global verifiability applied to voting:

> γ_ℓ contains all runs for which there exist choices of the dishonest voters (where a choice is either to abstain or to vote for one of the candidates) such that the result obtained together with the choices made by the honest voters in this run differs only by ℓ votes from the published result (i.e. the result that can be computed from the simple ballots on the bulletin board).

Another goal γ is presented in a case study [69, §6.2] of Helios:

> γ is satisfied in a run if the published result exactly reflects the actual votes of the honest voters in this run and votes of dishonest voters are distributed in some way on the candidates, possibly in a different way than how the dishonest voters actually voted.

These informal statements of goals are appealing, but they do not constitute rigorous mathematical definitions.[30] In our own work, we found that formal definitions were quite tricky to get right—for example, which ballots should be counted, how to count them, and how to determine whether that count differed from the published tally. So one contribution of our own work is to give a fully formal definition of election verifiability.

In their analysis of Helios, Küsters et al. [69] use goal γ to conclude that Helios 2.0 satisfies global verifiability. Yet Bernhard et al. [17] demonstrate an attack against the verifiability of Helios 2.0, and in the forthcoming technical report we show that Helios 2.0 does not satisfy Ver-Ext. This seeming discrepancy arises because the model in [69] does not formalize all the cryptographic primitives used by Helios, hence the attack goes unnoticed. So another contribution of our own work is to correctly distinguish between unverifiable and verifiable variants of Helios by rigorously analyzing the cryptography used in Helios.

[30]We shared [63] and discussed [64] our results with Küsters. In response, Küsters et al. propose a formalization of goals [62, §5.2]. We will consider this formalization in future work.

It is natural to ask whether election verifiability can be expressed in terms of global verifiability. We believe it can be. For instance, individual, universal and eligibility verifiability could be expressed, in the informal style of the goals quoted above, as the following goals:

- γ_{IV} is satisfied in a run if voters can uniquely identify their ballots on the bulletin board in this run.
- γ_{UV} is satisfied in a run if the correct tally of votes cast by authorized voters in this run is the same as the tally produced by algorithm Tally.
- γ_{EV} is satisfied in a run if every ballot tallied in this run was created by a voter in possession of a private credential.

We leave formalization of these goals as future work.

In the other direction, it is also natural to ask how global verifiability of γ_ℓ or γ would compare with election verifiability. Answering this question seems to require formalizing those goals. It would likely be possible to refine the informal statements of the goals into formal statements that are weaker, stronger, or even incomparable to our formal definition of election verifiability.

Küsters et al. [67] argue that deconstructing verifiability into individual and universal verifiability is insufficient to detect certain attacks involving ill-formed ballots. But those attacks leave open the possibility that there do exist notions of individual and universal verifiability that would be sufficient. Indeed, our own definition of universal verifiability rules out attacks based on ill-formed ballots, because *correct-tally* ensures that tallied ballots are well-formed.

VIII. Concluding Remarks

When we began this work, we were studying the Juels et al. definition of election verifiability [57]. We soon discovered that the definition fails to detect biasing and collusion attacks. While attempting to improve the Juels et al. definition to rule out those attacks, we discovered that factoring it into individual, universal, and eligibility verifiability led to an elegant decomposition of (mostly) orthogonal properties. We later sought to apply our new definitions to existing electronic voting systems, and Helios [5] and Civitas [28] were natural choices. But they treat authentication differently—Helios outsources authentication, whereas Civitas does not—so we were led to separate our definitions into variants for external and internal authentication. We were at first surprised to discover that JCJ, hence Civitas, does not satisfy the strong definition of eligibility verifiability. But upon reflection, it became apparent that an adversary who knows the tallier's private key can easily forge ballots that appear to be from eligible voters. Helios-C [32], however, avoids this problem by employing digital signatures.

Our definitions of verifiability have not addressed the issue of voter intent—that is, whether the ballot constructed by the

Vote algorithm corresponds to the candidate choice that a voter intended to make. Adida and Neff call this property "cast as intended" [6]. Many election schemes (e.g., [41], [49], [57]) do not satisfy cast as intended, because the schemes assume that voters construct ballots on trusted platforms. Nevertheless, schemes by Chaum [22], Neff [72], and Benaloh [10], [11] use cryptographic mechanisms to verify voter intent. It would be natural to explore strengthening our definitions to address voter intent.

The goal of this research is to enable verifiability of the voting systems we use in real-life, rather than merely trusting them. Research on verifiability can generalize beyond voting to other systems that must guarantee strong forms of integrity. Verifiable voting systems thus have the potential to contribute to the science of security, to democracy, and to broader society.

Acknowledgments

We thank David Bernhard, Jeremy Clark, Véronique Cortier, David Galindo, Markus Jakobsson, Steve Kremer, Ralf Küsters, Elizabeth Quaglia, Mark Ryan, Susan Thomson, and Poorvi Vora for insightful discussions which have influenced this paper. This work is partly supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project *CRYSP* (259639), by AFOSR grant FA9550-12-1-0334, and by the National Security Agency. This work was performed in part at George Washington University and INRIA.

References

[1] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2006.

[2] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[3] Ben Adida. Helios deployed at Princeton. http://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/ (accessed 7 May 2014), 2009.

[4] Ben Adida. Helios v4 Verification Specs. Helios documentation, http://documentation.heliosvoting.org/verification-specs/helios-v4 (accessed 2 May 2014), 2014. A snapshot of the specification on 8 Oct 2013 is available from https://web.archive.org/web/20131018033747/http://documentation.heliosvoting.org/verification-specs/helios-v4.

[5] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2009.

[6] Ben Adida and C. Andrew Neff. Ballot casting assurance. In *EVT'06: Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2006. USENIX Association.

[7] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08: 21st Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[8] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[9] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1996.

[10] Josh Benaloh. Simple Verifiable Elections. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[11] Josh Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[12] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.

[13] Josh Benaloh, Serge Vaudenay, and Jean-Jacques Quisquater. Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html (accessed 7 May 2014), Sept 2010.

[14] Josh Benaloh and Moti Yung. Distributing the Power of a Government to Enhance the Privacy of Voters. In *PODC'86: 5th Principles of Distributed Computing Symposium*, pages 52–62. ACM Press, 1986.

[15] David Bernhard. *Zero-Knowledge Proofs in Theory and Practice.* PhD thesis, Department of Computer Science, University of Bristol, 2014.

[16] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011.

[17] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

[18] Debra Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 http://www.sos.ca.gov/voting-systems/oversight/ttbr/db07-042-ttbr-system-decisions-release.pdf (accessed 7 May 2014), August 2007.

[19] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, October 2001.

[20] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *CSF'14: 27th Computer Security Foundations Symposium*. IEEE Computer Society, 2014. To appear.

[21] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[22] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.

[23] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Proc. Conference on Electronic Voting Technology*, pages 14:1–14:13. USENIX, 2008.

[24] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In *ESORICS'05: 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

[25] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Workshop on Trustworthy Elections*, 2006.

[26] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 191–199. Springer, 2010.

[27] Michael Clarkson, Brian Hay, Meador Inge, abhi shelat, David Wagner, and Alec Yasinsac. Software review and security analysis of Scytl remote voting software. Report commissioned by Florida Division of Elections. Available from http://election.dos.state.fl.us/voting-systems/pdf/FinalReportSept19.pdf. Filed September 19, 2008.

[28] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *S&P'08: 29th Security and Privacy Symposium*, pages 354–368. IEEE Computer Society, 2008.

[29] Josh Daniel Cohen and Michael J. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In *FOCS'85: 26th Symposium on Foundations of Computer Science*, pages 372–382. IEEE Computer Society, 1985.

[30] Véronique Cortier and David Galindo. Private communication, Nancy, France, 13th June 2013.

[31] Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachène. A generic construction for voting correctness at minimum cost - Application to Helios. Cryptology ePrint Archive, Report 2013/177 (version 20130521:145727), 2013.

[32] Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *ESORICS'14: 19th European Symposium on Research in Computer Security*, volume 8713 of *LNCS*, pages 327–344. Springer, 2014.

[33] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011.

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

[35] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[36] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[37] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[38] Jannik Dreier, Rosario Giustolisi, Ali Kassem, Pascal Lafourcade, and Gabriele Lenzini. On the Verifiability of (Electronic) Exams. Technical Report TR-2014-2, Verimag, 2014.

[39] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. Defining verifiability in e-auction protocols. In *ASIACCS'13: 8th ACM Symposium on Information, Computer and Communications Security*, pages 547–552. ACM Press, 2013.

[40] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[41] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[42] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO'01: 21st International Cryptology Conference*, volume 2139 of *LNCS*. Springer, 2001.

[43] David Galindo and Véronique Cortier. Private email communication, 19th June 2013.

[44] David Galindo and Véronique Cortier. Private email communication, Summer/Autumn 2014.

[45] Use of voting computers in 2005 Bundestag election unconstitutional, March 2009. Press release 19/2009 http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html (accessed 7 May 2014).

[46] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In *Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2004.

[47] Stuart Haber, Josh Benaloh, and Shai Halevi. The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/heliosDemo.pdf (accessed 7 May 2014), May 2010.

[48] Martin Hirt. Receipt-Free *K*-out-of-*L* Voting Based on ElGamal Encryption. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 64–82. Springer, 2010.

[49] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *EUROCRYPT'06: 25th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[50] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. In *WOTE'06: Workshop on Trustworthy Elections*, 2006.

[51] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. *Mathematical and Computer Modelling*, 48(9-10):1628–1645, 2008.

[52] IACR Elections. http://www.iacr.org/elections/ (accessed 7 May 2014), 2013.

[53] Markus Jakobsson and Ari Juels. Mix and Match: Secure Function Evaluation via Ciphertexts. In *ASIACRYPT'00: 6th International Conference on the Theory and Application of Cryptology and Information Security*, volume 1976 of *LNCS*, pages 162–177. Springer, 2000.

[54] Douglas W. Jones and Barbara Simons. *Broken Ballots: Will Your Vote Count?*, volume 204 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford University, 2012.

[55] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[56] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05: 4th Workshop on Privacy in the Electronic Society*, pages 61–70. ACM Press, 2005. See also http://www.colombia.rsa.com/rsalabs/staff/bios/ajuels/publications/Coercion/Coercion.pdf (accessed 7 May 2014).

[57] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 37–63. Springer, 2010.

[58] Shahram Khazaei and Douglas Wikström. Randomized partial checking revisited. In *Topics in Cryptology—CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 115–128. Springer Berlin Heidelberg, 2013.

[59] Aggelos Kiayias. Electronic voting. In *Handbook of Financial Cryptography and Security*, chapter 3. Chapman and Hall/CRC, 2010.

[60] Steve Kremer and Mark D. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *ESOP'05: 14th European Symposium on Programming*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[61] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10: 15th European Symposium on Research in Computer Security*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[62] Ralf Kuesters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211), 2015.

[63] Ralf Küsters. Private email communication, 24th June 2014.

[64] Ralf Küsters. Private email communication, October/November 2014.

[65] Ralf Küsters and Tomasz Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *S&P'09: 30th IEEE Symposium on Security and Privacy*, pages 251–266. IEEE Computer Society, 2009.

[66] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 526–535. ACM, 2010.

[67] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *S&P'11: 32nd IEEE Symposium on Security and Privacy*, pages 538–553. IEEE Computer Society, 2011. Full version available at http://infsec.uni-trier.de/publications/paper/KuestersTruderungVogt-TR-2011.pdf.

[68] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. *Journal of Computer Security*, 20(6):709–764, 2012.

[69] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *S&P'12: 33rd IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[70] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In *CRYPTO'06: 26th International Cryptology Conference*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.

[71] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *CCS'01: 8th ACM Conference on Computer and Communications Security*, pages 116–125. ACM Press, 2001.

[72] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Unpublished manuscript, 2004.

[73] Helios Princeton Elections. https://princeton.heliosvoting.org/ (accessed 7 May 2014), 2012.

[74] Participants of the Dagstuhl Conference on Frontiers of E-Voting. *Dagstuhl Accord*, 2007. http://www.dagstuhlaccord.org/ (accessed 7 May 2014).

[75] R. A. Peters. A secure bulletin board. Master's thesis, Technische Universiteit Eindhoven, June 2005.

[76] Kazue Sako and Joe Kilian. Secure Voting Using Partially Compatible Homomorphisms. In *CRYPTO'94: 14th International Cryptology Conference*, volume 839 of *LNCS*, pages 411–424. Springer, 1994.

[77] Daniel Sandler and Dan S. Wallach. Casting votes in the Auditorium. In *EVT'07: Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2007. USENIX Association.

[78] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Trans. Comput. Logic*, 6(3):634–671, July 2005.

[79] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.

[80] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *ESORICS'13: 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 463–480. Springer, 2013.

[81] Ben Smyth and David Bernhard. Ballot secrecy with malicious bulletin boards. Technical Report 2014/822, Cryptology ePrint Archive, 2014.

[82] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, volume 6186 of *LNCS*, pages 165–182. Springer, 2010.

[83] *Key issues and conclusions: May 2007 electoral pilot schemes*, May 2007. http://www.electoralcommission.org.uk/__data/assets/electoral_commission_pdf_file/0015/13218/Keyfindingsandrecommendationssummarypaper_27191-20111__E__N__S__W__.pdf (accessed 7 May 2014).

[84] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 441–457. Springer, 2013.