

# Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ

(Technical Report)

August 6, 2015

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 analyzed using these definitions. Helios 4.0 satisfies the definitions, but Helios 2.0 does not because of previously known attacks. JCJ does not satisfy the definitions because of a trust assumption it makes, but it does satisfy a weakened definition. Two previous definitions of verifiability (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 [22], [58], [76], [110]. So relying solely on trust in voting systems is unwise; verification of election outcomes is essential.<sup>1</sup>

*Election verifiability* enables voters and auditors to ascertain the correctness of election outcomes, regardless of whether the software and hardware of the voting system are trustworthy [1], [2], [31], [77], [100]. Kremer et al. [85] decompose election verifiability into three aspects:<sup>2</sup>

- *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.

We propose new definitions of these three aspects of verifiability in the computational model of cryptography. We show

that individual and universal verifiability are orthogonal, and that eligibility verifiability implies individual verifiability.

Because some electronic voting systems implement voter authentication themselves, whereas other systems outsource voter authentication to third parties, we develop two variants of our definitions—one for systems with *internal authentication* and another for systems with *external authentication*. We employ our definitions to analyze the verifiability of two well-known election schemes, JCJ [79] and Helios [5]. JCJ is an election scheme that achieves *coercion resistance* and has been implemented as Civitas [35]; it implements its own internal authentication. Helios is a web-based voting system that has been deployed in the real-world and outsources authentication.

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 [19], [20], [39], [40]. 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.

The JCJ election scheme does not satisfy our definition of eligibility verifiability, because an adversary who learns the tallier’s private key could cast unauthorized votes. We introduce a weakened definition of eligibility verifiability, incorporating JCJ’s trust assumption that the private key is unknown to the adversary, and show that JCJ satisfies our weakened definition of election verifiability with internal authentication.

Our definitions of election verifiability improve upon two previous definitions [38], [79] by detecting a new class of *collusion attacks*, in which the tallying algorithm announces an incorrect tally, and the verification algorithm colludes with the tallying algorithm to accept the incorrect tally. Examples of collusion attacks include vote stuffing, and announcing tallies that are independent of the election. Our definitions also improve upon those previous definitions by detecting

<sup>1</sup>*Doverlyai, no proveryai* (trust, but verify) says the Russian proverb.

<sup>2</sup>This decomposition has been criticized [91]; we refute that criticism in Section VII.

a new class of *biasing attacks*, in which the verification algorithm 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.

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

- proposing computational definitions of election verifiability,
- showing that individual, universal, and eligibility verifiability are mostly orthogonal properties of voting systems,
- 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 are not detected by two earlier definitions.

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

*Structure:* 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 introduces collusion and biasing attacks. Section VII reviews related work, and Section VIII concludes. Appendix A defines cryptographic primitives. The remaining appendices explore alternative definitions of verifiability, give the details of Helios and JCJ, and present proofs.

## 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.<sup>3</sup> 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 (Setup, Vote, Tally, Verify) of probabilistic polynomial-time (PPT) algorithms:

- **Setup**, denoted<sup>4</sup>  $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k)$ , is executed by the *tallier*, who is responsible for tallying ballots.<sup>5</sup> 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  $m_C$ .<sup>6</sup>

<sup>3</sup>[https://github.com/benadida/helios-server/tree/master/helios\\_auth/auth\\_systems](https://github.com/benadida/helios-server/tree/master/helios_auth/auth_systems), accessed 4 Aug 2015.

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

<sup>5</sup>Some election schemes (e.g., Helios and JCJ) permit the tallier’s role to be distributed amongst several talliers. For simplicity, we consider only a single tallier in this paper.

<sup>6</sup>The maximum ballots and candidate numbers are used to formalize Correctness. 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.

- **Vote**, denoted  $b \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$ , is executed by voters. A voter makes a *choice* of candidate from a sequence  $c_1, \dots, c_{n_C}$  of candidates. A *well-formed* choice is an integer  $\beta$ , 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  $\beta$  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  $(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k)$ , is executed by the tallier. It involves a public *bulletin board*  $BB$ , which we model as a set.<sup>7</sup> 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  $c_j$ .<sup>8</sup>
- **Verify**, denoted  $v \leftarrow \text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$ , 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 tallies produced by Tally corresponds to the choices input to Vote:

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

$$\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k);$$

**for**  $1 \leq i \leq n_B$  **do**

$$\left[ \begin{array}{l} b_i \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta_i, k); \\ \mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1; \end{array} \right.$$

$$BB \leftarrow \{b_1, \dots, b_{n_B}\};$$

$$(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k) :$$

$$n_B \leq m_B \wedge n_C \leq m_C \Rightarrow \mathbf{X} = \mathbf{Y} \rangle > 1 - \mu(k).$$

Note that Correctness does not involve an 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 definition of universal verifiability will ensure that, in the presence of an 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.

<sup>7</sup>Bulletin boards have also been modeled as public broadcast channels [43], [101], [103]. 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 [39], [40] demonstrate attacks against privacy when the bulletin board is modeled as a multiset.

<sup>8</sup>Let  $\mathbf{X}[i]$  denote component  $i$  of vector  $\mathbf{X}$ .

Election schemes must also satisfy Completeness, which stipulates that tallies produced by Tally will actually be accepted by Verify:

**Definition 2** (Completeness). *There exists a negligible function  $\mu$ , such that for all security parameters  $k$ , bulletin boards  $BB$ , and integers  $n_C$ , it holds that*

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

Without Completeness, election schemes might be vulnerable to biasing attacks, as we show in Section VI-B.

Finally, election schemes must satisfy Injectivity, which asserts that a ballot cannot be interpreted as a vote for more than one candidate:

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

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

Injectivity ensures that distinct choices are not mapped by Vote to the same ballot. Without Injectivity, an election scheme might produce ballots whose meaning is ambiguous. For example, if  $\text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r)$  were defined to be  $\beta + r$ , then a ballot  $b$  could be tallied as any well-formed choice  $\beta'$  such that  $\beta' = b - r'$  for some  $r'$ . But that definition of Vote is prohibited by Injectivity. Thus, Injectivity 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 Section 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 [31], MarkPledge [97], Scantegrity II [28], and Remotegrity [111] all rely on features implemented with paper, such as scratch-off surfaces and detachable columns.
- Everlasting privacy [95], 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 [34], 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 extension of our model to other 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 their ballot has been recorded, it suffices to enable them to uniquely identify their ballot on the bulletin board.<sup>9</sup>

Individual verifiability experiment  $\text{Exp-IV-Ext}(\Pi, \mathcal{A}, k)$ , where  $\Pi$  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 their ballot. In essence,  $\text{Exp-IV-Ext}$  challenges  $\mathcal{A}$  to generate a collision from Vote.<sup>10</sup> If  $\mathcal{A}$  cannot win, then voters can uniquely identify their ballots on the bulletin board:

```

Exp-IV-Ext( $\Pi, \mathcal{A}, k$ ) =
1 ( $PK_{\mathcal{T}}, n_C, \beta, \beta'$ )  $\leftarrow \mathcal{A}(k)$ ;
2  $b \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$ ;
3  $b' \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$ ;
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  $\beta$  and  $\beta'$ , such that ballots  $b$  and  $b'$  for those choices, as computed by Vote in lines 2 and 3, are equal. Individual verifiability thus resembles Injectivity, but individual verifiability allows choices to be equal and allows  $\mathcal{A}$  to choose election parameters.

One way to achieve individual verifiability is to base the election scheme on a probabilistic encryption scheme, such as El Gamal [51]. 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* [93], the adversary convinces some voters that a single ballot belongs to them all. Some clash attacks are possible because of vulnerabilities in the design of Vote. For example, if Vote simply outputs candidate choice  $\beta$ , then a voter has no way to distinguish their vote for  $\beta$  from another voter's vote for  $\beta$ .  $\text{Exp-IV-Ext}$  detects clash attacks resulting from vulnerabilities in Vote.

Some clash attacks, however, are possible because the adversary subverts the implementation of 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 algorithm  $\text{Vote}'$ . The conclusions drawn

<sup>9</sup>Section VIII addresses the complementary issue of whether a recorded ballot corresponds to the candidate choice a voter intended to make.

<sup>10</sup> $\text{Exp-IV-Ext}$  can be equivalently formulated as an experiment that challenges  $\mathcal{A}$  to predict the output of Vote. See Appendix B for details.

by a security analyst who uses our definition of individual verifiability to analyze `Vote` would not necessarily be applicable to `Vote'`.

In short, a voter can verify that their ballot has been recorded if and only if they run 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 use cut-and-choose protocols to audit ballots [13], [14]. 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  $\text{Exp-UV-Ext}(\Pi, \mathcal{A}, k)$  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  $\mathbf{Y} \leftarrow \text{correct-tally}(PK_{\mathcal{T}}, BB, n_C, k)$ ;
3 if  $\mathbf{X} \neq \mathbf{Y} \wedge \text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$  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. If  $\mathcal{A}$  cannot win  $\text{Exp-UV-Ext}$ , then `Verify` will not accept incorrect tallies. In particular, no ballots can be omitted from the tally, and at most one candidate choice can be included in the tally for each ballot.

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

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

The vector produced by *correct-tally* must be of length  $n_C$ . Component  $\beta$  of vector  $\text{correct-tally}(PK_{\mathcal{T}}, BB, n_C, k)$  equals  $\ell$  iff there exist<sup>11</sup>  $\ell$  ballots on the bulletin board that are votes for candidate  $\beta$ . It follows that the output of *correct-tally* represents the choices used to construct the recorded ballots. Note that, without Injectivity, the existential quantification in *correct-tally* could permit a ballot to be tallied for more than one candidate. Of course, *correct-tally* cannot be computed by a PPT algorithm for typical cryptographic election schemes. But that does not matter, because

<sup>11</sup>The definition of *correct-tally* employs a counting quantifier [105] denoted  $\exists^{\ell}$ . Predicate  $(\exists^{\ell} x : P(x))$  holds exactly when there are  $\ell$  distinct values for  $x$  such that  $P(x)$  is satisfied. Variable  $x$  is bound by the quantifier, whereas  $\ell$  is free.

*correct-tally* is never actually computed as part of an election scheme—its use is solely in the definition of  $\text{Exp-UV-Ext}$ .<sup>12</sup>

Security analysts must convince themselves that *correct-tally* is indeed correct. Because of the function's simplicity, this should be relatively straightforward. By comparison, Tally algorithms for real voting schemes tend to be complicated. For example, compare the complexity of *correct-tally* to Helios's Tally algorithm, which appears in Figure 1 of Appendix C.

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

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

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

### C. 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:*

- $\text{Setup}(k)$  outputs  $(\perp, \perp, p_1(k), p_2(k))$ , where  $p_1$  and  $p_2$  may be any polynomial functions.
- $\text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$  selects a nonce  $r$  uniformly at random from  $\mathbb{Z}_{2^k}$  and outputs  $(r, \beta)$ .

<sup>12</sup>Kiayias et al. [83] use a similar super-polynomial *vote extractor* to recover choices from ballots in an experiment defining verifiability.

- $\text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, n_C, k)$  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  $(\mathbf{X}, \perp)$ .
- $\text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$  outputs 1 if  $(\mathbf{X}, P) = \text{Tally}(\perp, \perp, BB, n_C, k)$  and 0 otherwise.

**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, Vote will select unique nonces for each voter, hence generate distinct ballots. Nonce also satisfies universal verifiability, because plaintext candidate choices are posted on the bulletin board.  $\square$

#### D. 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.<sup>13</sup> 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 [16], [65], [72], 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], [99].

Attacks have been discovered against the original Helios scheme, and defenses against those attacks have been proposed [19], [20], [39], [40]. For clarity, we write *Helios 2.0* to refer to the Helios scheme as originally proposed [5] and *Helios 4.0* to refer to a version of Helios that incorporates the defenses.<sup>14</sup> When referring in general to both of these schemes, we simply write *Helios*.

To achieve verifiability while maintaining *ballot secrecy* [18], [20], Helios homomorphically encrypts candidate choices. During tallying, all encrypted choices are homomorphically combined<sup>15</sup> into a single ciphertext, which is then

<sup>13</sup><https://vote.heliosvoting.org/>

<sup>14</sup>Our analysis of Helios 4.0 is based on the specification [4] for the next release. This specification incorporates proposals by Cortier and Smyth [40] for non-malleable ballots and by Bernhard et al. [20] to replace the weak Fiat–Shamir transformation with the strong Fiat–Shamir transformation.

<sup>15</sup>The homomorphic combination of ciphertexts is straightforward for two-candidate elections [12], [17], [36], [69], [102], since choices (e.g., “yes” or “no”) can be encoded as 1 or 0. Multi-candidate elections are also possible [17], [47], [68].

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.
- **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. During posting, the bulletin board 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 Appendix C.<sup>16</sup> Using that formalization, we can prove that Helios 4.0 is verifiable:

**Theorem 2.** *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.  $\square$

A formal proof of Theorem 2 appears in Appendix D. The proof assumes the random oracle model [9].

We would not expect Ver-Ext to hold for Helios 2.0, because of known attacks [20]. Accordingly, we prove that Helios 2.0 does not satisfy Ver-Ext in Appendix E.

### IV. INTERNAL AUTHENTICATION

Some election schemes implement their own authentication mechanisms. JCJ [77]–[79] and Civitas [35], 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.<sup>17</sup> 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.

<sup>16</sup>Our formalization is the first cryptographic description of Helios 4.0, hence an additional contribution of this work.

<sup>17</sup>Some election schemes (e.g., JCJ) permit the registrar’s role to be distributed among several registrars. For simplicity, we consider only a single registrar in this paper.

An *election scheme with internal authentication*, which henceforth in this section we abbreviate as “election scheme,” is a tuple (Setup, Register, Vote, Tally, Verify) of PPT algorithms. The algorithms are now denoted as follows:

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

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  $k$ , and 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.<sup>18</sup>

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

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

$$\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k);$$

**for**  $1 \leq i \leq n_B$  **do**

$(pk_i, sk_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k);$   
 $b_i \leftarrow \text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k);$   
 $\mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1;$

 $L \leftarrow \{pk_1, \dots, pk_{n_B}\};$   
 $BB \leftarrow \{b_1, \dots, b_{n_B}\};$   
 $(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k) :$   
 $n_B \leq m_B \wedge n_C \leq m_C \Rightarrow \mathbf{X} = \mathbf{Y} > 1 - \mu(k).$ 

**Definition 7** (Completeness). *There exists a negligible function  $\mu$ , such that for all security parameters  $k$ , bulletin boards  $BB$ , and integers  $n_C$  and  $n_V$ , it holds that*

$$\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k);$$

**for**  $1 \leq i \leq n_V$  **do**  $(pk_i, sk_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k);$

 $L \leftarrow \{pk_1, \dots, pk_{n_V}\};$   
 $(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k) :$   
 $|BB| \leq m_B \wedge n_C \leq m_C \Rightarrow$   
 $\text{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k) = 1 > 1 - \mu(k).$ 

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

$\beta'$ , we have

$$\Pr[(pk, sk) \leftarrow \text{Register}(PK_{\mathcal{T}}, k);$$

$$(pk', sk') \leftarrow \text{Register}(PK_{\mathcal{T}}, k);$$

$$b \leftarrow \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k);$$

$$b' \leftarrow \text{Vote}(sk', PK_{\mathcal{T}}, n_C, \beta', k) :$$

$$b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.$$

## 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.

In our experiments, below, we model an adversary who cannot corrupt the registration process that issues credentials to voters.<sup>19</sup> Hence our definitions will not detect attacks against verifiability that result solely from weaknesses in the registration process. Secure construction of electoral rolls is not a topic that electronic voting usually addresses—though it seems an important part of any real-world deployment.

1) *Individual verifiability*: The individual verifiability experiment again challenges adversary  $\mathcal{A}$  to generate a scenario in which the voter could not uniquely identify their ballot:<sup>20</sup>

$$\text{Exp-IV-Int}(\Pi, \mathcal{A}, k) =$$

- 1  $(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k);$
- 2 **for**  $1 \leq i \leq n_V$  **do**  $(pk_i, sk_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$
- 3  $L \leftarrow \{pk_1, \dots, pk_{n_V}\};$
- 4  $\text{Crpt} \leftarrow \emptyset;$
- 5  $(n_C, \beta, \beta', i, j) \leftarrow \mathcal{A}^C(L);$
- 6  $b \leftarrow \text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k);$
- 7  $b' \leftarrow \text{Vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta', k);$
- 8 **if**
- $b = b' \wedge b \neq \perp \wedge b' \neq \perp \wedge i \neq j \wedge sk_i \notin \text{Crpt} \wedge sk_j \notin \text{Crpt}$
- 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. The oracle is used to model  $\mathcal{A}$  corrupting voters and learning their private credentials: on invocation  $C(\ell)$ , where  $1 \leq \ell \leq n_V$ , the oracle records that voter  $\ell$  is corrupted by updating  $\text{Crpt}$  to be  $\text{Crpt} \cup \{sk_{\ell}\}$  and outputs  $sk_{\ell}$ . In line 5, the voter indices output by  $\mathcal{A}$  must be legal with respect to  $n_V$ , but we elide that detail from the experiment for simplicity. Line 8 ensures that  $\mathcal{A}$  cannot trivially win by corrupting voters.

<sup>18</sup>It might 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., [35], [79], the registrar does not sign credential pairs with its own private key. Rather, the registrar signs the electoral roll.

<sup>19</sup>Küsters and Truderung [89] explore some consequences of permitting adversarial influence during registration.

<sup>20</sup>Unlike Exp-IV-Ext, a variant of Exp-IV-Int that challenges  $\mathcal{A}$  to predict the output of Vote is strictly stronger. See Appendix B for details.

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

```

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 ( $pk_i, sk_i$ )  $\leftarrow$  Register( $PK_{\mathcal{T}}, k$ )
3  $L \leftarrow \{pk_1, \dots, pk_{n_V}\}$ ;
4  $M \leftarrow \{(pk_1, sk_1), \dots, (pk_{n_V}, sk_{n_V})\}$ ;
5 ( $BB, n_C, \mathbf{X}, P$ )  $\leftarrow \mathcal{A}(M)$ ;
6  $\mathbf{Y} \leftarrow$  correct-tally( $PK_{\mathcal{T}}, BB, M, n_C, k$ );
7 if  $\mathbf{X} \neq \mathbf{Y} \wedge$  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  $PK_{\mathcal{T}}, BB, M, n_C, k, \ell$ , and  $\beta \in \{1, \dots, n_C\}$ ,

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

Let *authorized* be defined as follows:

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

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 be 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.

Eligibility verifiability experiment Exp-EV-Int 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 ( $pk_i, sk_i$ )  $\leftarrow$  Register( $PK_{\mathcal{T}}, k$ );
3  $L \leftarrow \{pk_1, \dots, pk_{n_V}\}$ ;
4  $Crpt \leftarrow \emptyset$ ;  $Rvld \leftarrow \emptyset$ ;
5 ( $n_C, \beta, i, b$ )  $\leftarrow \mathcal{A}^{C,R}(L)$ ;
6 if  $\exists r : b = \text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp \wedge b \notin$ 
   $Rvld \wedge sk_i \notin Crpt$  then
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. 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 models  $\mathcal{A}$  coercing voters to reveal their private credentials. The latter set models  $\mathcal{A}$  observing ballots 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:

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

In line 6,  $\mathcal{A}$  wins if (i) the ballot is *authentic*, meaning that it is the output of *Vote* on an authorized credential, and (ii) that credential belongs to a voter that  $\mathcal{A}$  did not corrupt, and (iii) that ballot was not revealed. If  $\mathcal{A}$  cannot succeed in this experiment, then 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  $\Pi$  satisfies election verifiability with internal authentication (Ver-Int) if for all PPT adversaries  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all security parameters  $k$ , it holds that  $\text{Succ}(\text{Exp-IV-Int}(\Pi, \mathcal{A}, k)) + \text{Succ}(\text{Exp-UV-Int}(\Pi, \mathcal{A}, k)) + \text{Succ}(\text{Exp-EV-Int}(\Pi, \mathcal{A}, k)) \leq \mu(k)$ .*

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

### C. Example—Toy schemes from digital signatures

A toy election scheme satisfying Ver-Int can be based on a digital signature scheme (Gen, Sign, Ver).<sup>21</sup> Each voter publishes their signed candidate choice on the bulletin board.

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

- Setup( $k$ ) outputs  $(\perp, \perp, p_1(k), p_2(k))$ , where  $p_1$  and  $p_2$  may be any polynomial functions.
- Register( $PK_{\mathcal{T}}, k$ ) computes  $(pk, sk) \leftarrow \text{Gen}(1^k)$  and outputs  $(pk, sk)$ .
- Vote( $sk, PK_{\mathcal{T}}, n_C, \beta, k$ ) outputs  $(\beta, \text{Sign}(sk, \beta))$ .
- 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  $(\mathbf{X}, \perp)$ .
- Verify( $PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k$ ) outputs 1 if  $(\mathbf{X}, P) = \text{Tally}(\perp, \perp, BB, L, n_C, \perp)$  and 0 otherwise.

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

**Proposition 3.** If (Gen, Sign, Ver) is a signature scheme satisfying existential unforgeability under adaptive chosen-message attack,<sup>22</sup> 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.  $\square$

### D. Orthogonality

Exp-IV-Int, Exp-UV-Int, and Exp-EV-Int capture mostly orthogonal security properties, as shown in Table I. Individual and universal verifiability are orthogonal, and eligibility verifiability implies individual verifiability.

**Theorem 4.** If an election scheme  $\Pi$  satisfies Exp-EV-Int, then  $\Pi$  also satisfies Exp-IV-Int.

*Proof sketch.* If  $\Pi$  satisfies Exp-EV-Int, 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 their ballot, because no one else knows their private credential. Therefore  $\Pi$  satisfies Exp-IV-Int.  $\square$

The proof of Theorem 4 appears in Appendix F.

In Table I, AlwaysVerify( $\cdot$ ) is a function that transforms an election scheme by compromising Verify to always return 1. Thus, AlwaysVerify( $\Pi$ ) is guaranteed not to satisfy Exp-UV-Int. Similarly, 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

Line	IV	UV	EV	Scheme
1	$\times$	$\times$	$\times$	AlwaysVerify(IgnoreCreds(Choice))
2	$\times$	$\times$	$\checkmark$	—
3	$\times$	$\checkmark$	$\times$	IgnoreCreds(Choice)
4	$\times$	$\checkmark$	$\checkmark$	—
5	$\checkmark$	$\times$	$\times$	AlwaysVerify(IgnoreCreds(Nonce))
6	$\checkmark$	$\times$	$\checkmark$	AlwaysVerify(Sig)
7	$\checkmark$	$\checkmark$	$\times$	Malleable Sig
8	$\checkmark$	$\checkmark$	$\checkmark$	Sig

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

credentials altogether: Register returns  $(\perp, \perp)$ , Vote ignores  $sk$ , and Tally and Verify ignore  $L$ . Thus, 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-D) that Choice is the election scheme in which ballots contain only the plaintext candidate choice. By compromising Verify and ignoring credentials, we obtain a scheme that satisfies no properties.
- 2) By Theorem 4, this situation is impossible.
- 3) Compared to line 1 of Table I, this scheme satisfies Exp-UV-Int, because Verify is not compromised.
- 4) By Theorem 4, this situation is impossible.
- 5) Nonce 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,<sup>23</sup> 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

JCJ (named for its designers, Juels, Catalano, and Jakobson) [77]–[79] 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.<sup>24</sup> 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:

<sup>23</sup>Given a message  $m$  and signature  $\sigma$ , a *malleable* signature scheme permits computation of a signature  $\sigma'$  on a related message  $m'$  [25]. 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  $\beta$  into a signature on a distinct, well-formed candidate  $\beta'$ .

<sup>24</sup>Chaum [26] introduced mixnets. Adida [1] surveys verifiable mixnets.

<sup>21</sup>Digital signature schemes are defined in Appendix A.

<sup>22</sup>This security property is defined in Appendix A.

- **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. 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) [74]. (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.

Appendix G 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 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  $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k)$ ;
2  $n_V \leftarrow \mathcal{A}(PK_{\mathcal{T}}, k)$ ;
3 for  $1 \leq i \leq n_V$  do  $(pk_i, sk_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ ;
4  $L \leftarrow \{pk_1, \dots, pk_{n_V}\}$ ;
5  $Crpt \leftarrow \emptyset$ ;  $Rvld \leftarrow \emptyset$ ;
6  $(n_C, \beta, i, b) \leftarrow \mathcal{A}^{C,R}(L)$ ;
7 if  $\exists r : b = \text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp \wedge b \notin$ 
    $Rvld \wedge sk_i \notin Crpt$  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  $\Pi$  satisfies weak election verifiability with internal authentication (Ver-Int-Weak) if for all probabilistic polynomial-time adversaries  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all security parameters  $k$ , we have  $\text{Succ}(\text{Exp-IV-Int}(\Pi, \mathcal{A}, k)) + \text{Succ}(\text{Exp-UV-Int}(\Pi, \mathcal{A}, k)) + \text{Succ}(\text{Exp-EV-Int-Weak}(\Pi, \mathcal{A}, k)) \leq \mu(k)$ .*

**Theorem 5.** *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.  $\square$

A formal proof of Theorem 5 appears in Appendix H. The proof assumes the random oracle model.

The Civitas [35] scheme refines the JCJ scheme. Some refinements relevant to election verifiability are an implementation of a distributed registration protocol, and a mixnet based on randomized partial checking (RPC) [75]. We leave a proof that Civitas satisfies Ver-Int-Weak as future work. In that proof, it would be necessary to assume the RPC construction satisfies the definition of mixnets given in Appendix A. Work by Khazaei and Wikström [81] suggests that actually proving satisfaction is unlikely to be possible. Alternatively, the mixnet could be replaced by one based on zero-knowledge proofs [54], [96].

## 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 algorithms might be designed such that they collude to accept incorrect tallies.
- *Biasing attacks.* An election scheme’s verification algorithm might be designed such that it rejects some legitimate tallies.

Although a well-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 errors. So definitions of election verifiability should preclude collusion and biasing attacks.

### A. Collusion Attacks

Here are two examples of potential collusion attacks:

- **Vote stuffing.** Tally behaves normally, but adds  $\kappa$  votes for candidate  $\beta$ . Verify subtracts  $\kappa$  votes from  $\beta$ , then

proceeds with verification as normal. Elections thus verify as normal, except that candidate  $\beta$  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.

Vote 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. Note that vote stuffing is not a failure of eligibility verifiability, because the stuffed votes do not correspond to any 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.

We show, next, that the definition of election verifiability by Juels et al. [79] fails to detect vote stuffing and backdoor tally replacement, and that the definition by Cortier et al. [38] fails to detect backdoor tally replacement.

Juels et al. [79] formalize definitions that we name *JCJ-correctness* and *JCJ-verifiability*. JCJ-correctness is intuitively 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” [79, p. 45]; it is formalized in terms of whether the adversary can post ballots on the bulletin board that cause the tally to be computed incorrectly. JCJ-verifiability is intuitively “the ability for any player to check whether the tally... has been correctly computed” [79, p. 46]; it is formalized in terms of whether Verify will accept a tally that differs from the output of Tally. We restate the formal definitions in Appendix I.

To show that the JCJ definitions fail to detect collusion attacks, we first formalize the vote stuffing attack. An election scheme  $\Pi = (\dots, \text{Tally}, \text{Verify})$  can be modified to derive a vote-stuffing election scheme  $\text{Stuff}(\Pi, \beta, \kappa) = (\dots, \text{Tally}_S, \text{Verify}_S)$ , which adds  $\kappa$  votes to  $\beta$ , as follows:<sup>25</sup>

- $\text{Tally}_S(\dots)$  computes  $(\mathbf{X}, P) \leftarrow \text{Tally}(\dots)$  and outputs  $(\text{Add}(\mathbf{X}, \beta, \kappa), P)$ .<sup>26</sup>
- $\text{Verify}_S(\dots, \mathbf{X}, P, \dots)$  computes  $v \leftarrow \text{Verify}(\dots, \text{Add}(\mathbf{X}, \beta, -\kappa), P, \dots)$  and outputs  $v$ .

**Proposition 6.** *Given an election scheme  $\Pi$  satisfying JCJ-correctness and JCJ-verifiability, and integers  $\beta$  and  $\kappa$ , it holds that  $\text{Stuff}(\Pi, \beta, \kappa)$  satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix J.

Second, we formalize the backdoor tally replacement attack. An election scheme  $\Pi = (\dots, \text{Tally}, \text{Verify})$  can be modified

<sup>25</sup>We omit many of the parameters of Tally and Verify here for simplicity; see Appendix J for details.

<sup>26</sup>Let  $\text{Add}(\mathbf{X}, \beta, \kappa) = (\mathbf{X}[1], \dots, \mathbf{X}[\beta - 1], \mathbf{X}[\beta] + \kappa, \mathbf{X}[\beta + 1], \dots, \mathbf{X}[|\mathbf{X}|])$ . And let  $|\mathbf{X}|$  denote the length of vector  $\mathbf{X}$ .

to derive an election scheme  $\text{Backdoor}(\Pi) = (\dots, \text{Tally}_B, \text{Verify}_B)$ , as follows:

- $\text{Tally}_B(\dots, SK_{\mathcal{T}}, BB, \dots)$  checks whether there exists a unique  $\mathbf{X}^*$  such that  $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$ . If so,  $\text{Tally}_B$  outputs  $(\mathbf{X}^*, \perp)$ . Otherwise,  $\text{Tally}_B$  computes  $(\mathbf{X}, P) \leftarrow \text{Tally}(\dots, SK_{\mathcal{T}}, BB, \dots)$  and outputs  $(\mathbf{X}, P)$ .
- $\text{Verify}_B(PK_{\mathcal{T}}, \dots, BB, \dots, \mathbf{X}, P, \dots)$  checks whether there exists a unique  $\mathbf{X}^*$  such that  $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$ .<sup>27</sup> If so,  $\text{Verify}_B$  outputs 1. Otherwise,  $\text{Verify}_B$  computes  $v \leftarrow \text{Verify}(PK_{\mathcal{T}}, \dots, BB, \dots, \mathbf{X}, P, \dots)$  and outputs  $v$ .

**Proposition 7.** *Given an election scheme  $\Pi$  satisfying JCJ-correctness and JCJ-verifiability that does not leak the tallier’s private key, it holds that  $\text{Backdoor}(\Pi)$  satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix J, where we also formally define key leakage.

Cortier et al. [38] propose definitions similar to *JCJ-verifiability* and insist that election schemes must satisfy their notions of correctness and partial tallying. Vote 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 [37], [55], [56].

### B. Biasing attacks

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

- **Reject All.** Let  $\text{Reject}(\Pi)$  be  $(\dots, \text{Verify}_R)$ , where  $\text{Verify}_R$  always outputs 0.  $\text{Verify}_R$  therefore always rejects, hence no election can ever be considered valid.
- **Selective Reject.** Let  $\varepsilon$  be a distinguished value that would not be posted on the bulletin board by honest voters. Let  $\text{Selective}(\Pi, \varepsilon)$  be  $(\dots, \text{Verify}_R)$ , where  $\text{Verify}_R(\dots, BB, \dots)$  computes  $v \leftarrow \text{Verify}(\dots, BB, \dots)$  and outputs 1 if both  $v = 1$  and  $\varepsilon \notin BB$ . Otherwise,  $\text{Verify}_R$  outputs 0.  $\text{Verify}_R$  therefore rejects if  $\varepsilon$  appears on the bulletin board, hence some elections can be invalidated.
- **Biased Reject.** Suppose  $Z$  is a set of tallies. Let  $\text{Bias}(\Pi, Z)$  be  $(\dots, \text{Verify}_R)$ , where  $\text{Verify}_R(\dots, \mathbf{X}, \dots)$  computes  $v \leftarrow \text{Verify}(\dots, \mathbf{X}, \dots)$  and outputs 1 if both  $v = 1$  and  $\mathbf{X} \in Z$ . Otherwise,  $\text{Verify}_R$  outputs 0.  $\text{Verify}_R$  therefore only accepts a subset of the tallies accepted by Verify, hence biases tallies toward  $Z$ .

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

The definition of verifiability by Juels et al. [79] fails to detect all three of the above attacks, because that definition has no notion of Completeness. For example, it is vulnerable to Biased Reject attacks:

<sup>27</sup> $\text{Verify}_B$  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.

**Proposition 8.** *Given an election scheme  $\Pi$  satisfying JCJ-correctness and JCJ-verifiability, and given a multiset  $Z$ , it holds that  $\text{Bias}(\Pi, Z)$  satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix J.

The definition of verifiability by Kiayias et al. [83] fails to detect Selective Reject attacks, because (like JCJ) the definition has no notion of Completeness. Their notion of Correctness does rule out Reject All and Biased Reject attacks.

Similarly, the definition of verifiability by Cortier et al. [38] detects Biased Reject and Reject All attacks, but fails to detect Selective Reject attacks, because that definition’s notion of Completeness does not quantify over all bulletin boards.

## VII. RELATED WORK

Kiayias [82] 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. [77]–[79] and Cortier et al. [38] give game-based definitions of verifiability. 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 [15], 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 [36], 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.

Kiayias et al. [83] define a property they name *E2E verifiability* (E2E abbreviates “end-to-end”). This property combines our intuitive notions of individual and universal verifiability into a single definition. Their definition fails to detect Selective Reject attacks (cf. §VI). Their definitions, like ours, do not address voter intent—that is, verification by humans that ballots correctly encode candidate choices—as we discuss in Section VIII.

Also in the computational model, Groth [63], and Moran and Naor [95], state definitions of verifiability in terms of *universal composability* [23]. 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 [63, p. 2], but Moran and Naor’s does [95, p. 386].

In the symbolic model, Smyth et al. [109] 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. [85] overcome this limitation with a weaker definition that sacrifices amenability to automated reasoning, and Smyth [106, §3] extends this definition. Dreier et al. have adapted election verifiability to auction [50] and examination [49] systems.

Also in the symbolic model, Kremer and Ryan [84] and Backes et al. [7] formalize definitions of *eligibility*. 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 [84, §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. [85], [106], [109] 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 cryptographic election schemes assume it, too. Ways to construct secure bulletin boards have been proposed, e.g., [45], [67], [101], [103]. We postpone a discussion of cast as intended to Section VIII.

Privacy properties [48], [79], [91], [92], [107], [108]—such as ballot secrecy, receipt freeness, and coercion resistance—complement verifiability. Chevallier-Mames et al. [32], [33] and Hosp and Vora [70], [71] 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 2 and 5 coupled with existing privacy results such as the ballot secrecy proofs for Helios 4.0 [20, Theorem 3], [18, Theorem 6.12], and the coercion resistance proof for JCJ [79, §5].

*Comparison with global verifiability*: Küsters et al. [90], [91], [93] 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  $\gamma_\ell$  is presented in a case study [91, §5.2] of global verifiability applied to voting:

$\gamma_\ell$  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  $\ell$  votes

from the published result (i.e. the result that can be computed from the simple ballots on the bulletin board).

Another goal  $\gamma$  is presented in a case study [93, §6.2] of Helios:

$\gamma$  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. As Kiayias et al. write, “[global verifiability] has the disadvantage that the set  $\gamma$  remains undetermined and thus the level of verifiability that is offered by the definition hinges on the proper definition of  $\gamma$  which may not be simple” [83, p. 476]. 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 we shared [87] and discussed [88] our results with Küsters. In response, Küsters et al. updated an online technical report to propose a formalization of goals [86, §5.2]; we look forward to analyzing that formalization when it is published.

In an analysis of Helios, Küsters et al. [93] use goal  $\gamma$  to conclude that Helios 2.0 satisfies global verifiability. Yet Bernhard et al. [20] demonstrate an attack against the verifiability of Helios 2.0, and in Appendix E we show that Helios 2.0 does not satisfy Ver-Ext. This seeming discrepancy arises because the analysis in [93] 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.

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:

- $\gamma_{IV}$  is satisfied in a run if voters can uniquely identify their ballots on the bulletin board in this run.
- $\gamma_{UV}$  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.
- $\gamma_{EV}$  is satisfied in a run if every ballot tallied in this run was created by a voter in possession of a private credential.

Küsters et al. [91] 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.

One concern that might be raised is whether there still lurk any “gaps” in our decomposition into individual and universal (and eligibility) verifiability. Indeed, there might be. But the definition of global verifiability does not rule out the possibility of gaps, either: any gap in the formal statement of a goal will lead to a vulnerability. That is, if the analyst forgets to include some necessary facet of verifiability when stating the formal goal, then global verifiability will not detect any attacks against that facet. Global verifiability does not guarantee a lack of gaps.

## VIII. CONCLUDING REMARKS

When we began this work, we were studying the Juels et al. [79] definition of election verifiability. We 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 [35] 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.

Our definitions of verifiability have not addressed the issue of voter intent—that is, verification by a human that the ballot submitted by a voter corresponds to the candidate choice the voter intended to make. Adida and Neff call this property “cast as intended” [6]. Many election schemes (e.g., [53], [69], [79], [83]) do not satisfy cast as intended, because the schemes implicitly or explicitly assume that voters can themselves verify the cryptographic operations required to construct ballots. Nevertheless, schemes by Chaum [27], Neff [97], and Benaloh [13], [14] introduce 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 that 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 grants FA9550-12-1-0334 and FA9550-14-1-0334, by NSF grant 1421373, and by the National Security Agency. This work was performed in part at George Washington University and INRIA.

#### DEDICATION<sup>28</sup>

Ben Smyth dedicates his contribution to the loving memory of Anne Konishi, 1971 – 2015. What matters most of all is the dash. We had a great time.

He writes for Christina Mai Konishi. Smile like your mother, for good fortune seeks those who smile (*warau kado niwa fuku kitaru*, says the Japanese proverb).

#### APPENDIX A CRYPTOGRAPHIC PRIMITIVES

##### A. Basic definitions

**Definition 12** (Negligible function [59]). A function  $\mu : \mathbb{N} \rightarrow \mathbb{R}$  is negligible if for every positive polynomial  $p(\cdot)$ , there exists an  $N$ , such that for all  $n > N$ ,

$$\mu(n) < \frac{1}{p(n)}.$$

An event  $E(k)$ , where  $k$  is a security parameter, occurs with *negligible probability* if  $\Pr[E(k)] \leq \mu(k)$  for some negligible function  $\mu$ . The event occurs with *overwhelming probability* if the complement of the event occurs with negligible probability.

**Definition 13** (Asymmetric encryption scheme [80]). An asymmetric encryption scheme is a tuple of PPT algorithms  $(\text{Gen}, \text{Enc}, \text{Dec})$  such that:

- **Gen**, denoted  $(pk, sk, \mathfrak{m}) \leftarrow \text{Gen}(1^k)$ , takes a security parameter  $1^k$  as input and outputs a key pair  $(pk, sk)$  and message space  $\mathfrak{m}$ .
- **Enc**, denoted  $c \leftarrow \text{Enc}(pk, m)$ , takes a public key  $pk$  and message  $m \in \mathfrak{m}$  as input, and outputs a ciphertext  $c$ .
- **Dec**, denoted  $m \leftarrow \text{Dec}(pk, sk, c)$ , takes a public key  $pk$ , a private key  $sk$ , and ciphertext  $c$  as input, and outputs a message  $m$  or error symbol  $\perp$ . We assume Dec is deterministic.

Moreover, the scheme must be correct: there exists a negligible function  $\mu$ , such that for all security parameters  $k$  and messages  $m$ , we have  $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \text{Gen}(1^k); c \leftarrow \text{Enc}(pk, m) : m \in \mathfrak{m} \Rightarrow \text{Dec}(pk, sk, c) = m] > 1 - \mu(k)$ .

Our definition of asymmetric encryption schemes differs from Katz and Lindell’s definition [80, Definition 10.1] in that we formally state the plaintext space, and we provide the public key as input to Dec. The latter is a technical convenience that we use to handle parameters needed for encryption schemes. For example, El Gamal is defined in terms of a cyclic group, and a description of the group parameters is needed to compute encryptions and decryptions. We assume those parameters are encoded as part of the public key. Although we could also

assume the parameters are encoded in private keys, it suffices to pass the public key into the decryption algorithm.

**Definition 14** (Homomorphic encryption [80]). An asymmetric encryption scheme  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$  is homomorphic if for all  $k, pk, sk$  and  $\mathfrak{m}$ , such that  $(pk, sk, \mathfrak{m}) \leftarrow \text{Gen}(1^k)$ , there exist binary operators  $\odot, \oplus$  and  $\otimes$ , and sets  $\mathfrak{c}$  and  $\mathfrak{r}$ , such that  $(\mathfrak{m}, \odot)$ ,  $(\mathfrak{r}, \oplus)$  and  $(\mathfrak{c}, \otimes)$  are groups and

- For all  $m$  and  $c$ , such that  $m \in \mathfrak{m}$  and  $c \leftarrow \text{Enc}(pk, m)$ , it holds that  $c \in \mathfrak{c}$ .
- For all  $m_1, m_2 \in \mathfrak{m}$  and  $c_1, c_2 \in \mathfrak{c}$ , such that  $\text{Dec}(pk, sk, c_1) = m_1$  and  $\text{Dec}(pk, sk, c_2) = m_2$ , there exists a negligible function  $\mu$ , such that  $\Pr[\text{Dec}(pk, sk, c_1 \otimes c_2) = m_1 \odot m_2] > 1 - \mu(k)$ .
- For all  $m_1, m_2 \in \mathfrak{m}$  and  $r_1, r_2 \in \mathfrak{r}$ , there exists a negligible function  $\mu$ , such that  $\Pr[\text{Enc}(pk, m_1; r_1) \otimes \text{Enc}(pk, m_2; r_2) = \text{Enc}(pk, m_1 \odot m_2; r_1 \oplus r_2)] > 1 - \mu(k)$ .

The scheme  $\Gamma$  is *additively homomorphic* if  $\odot$  is the addition operator in  $\mathfrak{m}$ ; or, *multiplicatively homomorphic* if  $\odot$  is the multiplication operator in  $\mathfrak{m}$ .

Our definition of homomorphic encryption strengthens Katz and Lindell’s definition [80, Definition 11.35] by adding the third bullet point, which requires the homomorphism to extend to random coins. That extension is needed in Helios as part of the Vote algorithm, to enable proofs of plaintext knowledge on homomorphic combinations of ciphertexts.

*Indistinguishability under chosen-plaintext attack* (IND-CPA) [8], [10], [11], [60], [61] is a standard definition of security for encryption schemes. Intuitively, if an encryption scheme satisfies IND-CPA, then an adversary without access to a decryption oracle is unable to distinguish ciphertexts. A variant (IND- $j$ -CPA) allows the adversary  $j$  adaptive queries to a decryption oracle, where each query is a *parallel decryption query*—i.e., it requests the decryption of a vector of ciphertexts. Hence, IND-0-CPA is equivalent to IND-CPA.

**Definition 15** (IND- $j$ -CPA [21]). An asymmetric encryption scheme  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$  satisfies *IND- $j$ -CPA* if for all stateful PPT adversaries  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all security parameters  $k$ , we have  $\text{Succ}(\text{Exp-CPA}(j, \Gamma, \mathcal{A}, k)) \leq \frac{1}{2} + \mu(k)$ , where  $j$  is a non-negative integer and the experiment Exp-CPA is defined as follows<sup>29</sup>:

$$\text{Exp-CPA}(j, \Gamma, \mathcal{A}, k) =$$

<sup>29</sup>Let  $x \leftarrow_R S$  denote assignment to  $x$  of an element chosen uniformly at random from set  $S$ .

<sup>28</sup>The dedication references Linda Ellis (1996) *The Dash*.

```

1  $(pk, sk, m) \leftarrow \text{Gen}(1^k);$ 
2  $(m_0, m_1) \leftarrow \mathcal{A}(pk, m);$ 
3  $b \leftarrow_R \{0, 1\};$ 
4  $c \leftarrow \text{Enc}(pk, m_b);$ 
5  $b' \leftarrow \mathcal{A}^\mathcal{O}(c);$ 
6 if  $b = b' \wedge m_0, m_1 \in \mathbf{m} \wedge |m_0| = |m_1|$  then
7   | return 1
8 else
9   | return 0

```

where  $\mathcal{A}$  has access to a decryption oracle  $\mathcal{O}$ , which is defined as follows<sup>30</sup>:

```

 $\mathcal{O}(c) =$ 
1 if  $j > 0 \wedge \bigwedge_{1 \leq i \leq |c|} c \neq \mathbf{c}[i]$  then
2   |  $j \leftarrow j - 1;$ 
3   | return  $(\text{Dec}(pk, sk, \mathbf{c}[1]), \dots, \text{Dec}(pk, sk, \mathbf{c}[|c|]))$ 
4 else
5   | return  $\perp$ 

```

**Definition 16** (Signature scheme [80]). A signature scheme is a tuple  $(\text{Gen}, \text{Sign}, \text{Ver})$  of PPT algorithms such that:

- **Gen**, denoted  $(pk, sk) \leftarrow \text{Gen}(1^k)$ , takes a security parameter  $1^k$  as input and outputs a key pair  $(pk, sk)$ .
- **Sign**, denoted  $\sigma \leftarrow \text{Sign}(sk, m)$ , takes a private key  $sk$  and message  $m$  as input, and outputs a signature  $\sigma$ .
- **Verify**, denoted  $v \leftarrow \text{Ver}(pk, m, \sigma)$ , takes a public key  $pk$ , message  $m$ , and signature  $\sigma$  as input, and outputs a bit  $v$ , which is 1 if the signature successfully verifies and 0 otherwise. We assume  $\text{Ver}$  is deterministic.

Moreover, the scheme must be correct: there exists a negligible function  $\mu$ , such that for all security parameters  $k$  and messages  $m$ , we have  $\Pr[(pk, sk) \leftarrow \text{Gen}(1^k); \sigma \leftarrow \text{Sign}(sk, m); \text{Ver}(pk, m, \sigma) = 1] > 1 - \mu(k)$ .

**Definition 17** (EU-CMA [80]). A signature scheme  $\Gamma = (\text{Gen}, \text{Sign}, \text{Ver})$  satisfies existential unforgeability under adaptive chosen-message attack (EU-CMA) if for all PPT adversaries  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all security parameters  $k$ , we have  $\text{Succ}(\text{Exp-Sign}(\Gamma, \mathcal{A}, k)) \leq \mu(k)$ , where experiment  $\text{Exp-Sign}$  is defined as follows:

```

 $\text{Exp-Sign}(\Gamma, \mathcal{A}, k) =$ 
1  $(pk, sk) \leftarrow \text{Gen}(1^k);$ 
2  $\text{Msg} \leftarrow \emptyset;$ 
3  $(m, \sigma) \leftarrow \mathcal{A}^\mathcal{O}(pk, 1^k);$ 
4 if  $\text{Ver}(pk, m, \sigma) = 1 \wedge m \notin \text{Msg}$  then
5   | return 1
6 else
7   | return 0

```

The experiment defines an oracle  $\mathcal{O}$ . On invocation  $\mathcal{O}(m)$ , oracle  $\mathcal{O}$  computes a signature  $\sigma \leftarrow \text{Sign}(sk, m)$ , records that

<sup>30</sup>The oracle in experiment  $\text{Exp-CPA}$  may access parameter  $j$ . Henceforth, we continue to allow oracles to access experiment parameters without explicitly mentioning them.

the adversary requested a signature on  $m$  by updating  $\text{Msg}$  to be  $\text{Msg} \cup \{m\}$ , and outputs  $\sigma$ .

## B. Proof systems

A *proof system* (originally known as an *interactive proof system* [62]) is a two-party protocol between a prover and a verifier. The prover convinces the verifier that a string  $x$  is in a language  $L$ . Here, we assume that there is a *witness relation*  $R$ , such that  $s \in L$  iff there exists a witness  $w$ , such that  $(s, w) \in R$ . For any  $(s, w) \in R$ , it must also hold that the length of  $w$  is at most polynomial in the length of  $s$ . Proof systems ensure that a prover can convince a verifier of any valid claim (*completeness*), and that a verifier cannot be fooled into accepting a false claim (*soundness*).

A *sigma protocol* [46], [66] is a proof system with a particular three-move structure: commit, challenge, respond.

**Definition 18** (Sigma protocol). A sigma protocol for a relation  $R$  is a tuple  $(\text{Comm}, \text{Chal}, \text{Resp}, \text{Verify})$  of PPT algorithms such that:

- **Comm**, denoted  $(\text{comm}, t) \leftarrow \text{Comm}(s, w)$ , is executed by a prover.  $\text{Comm}$  takes a statement  $s$  and witness  $w$  as input, and outputs a commitment  $\text{comm}$  and some state information  $t$ .
- **Chal**, denoted  $\text{chal} \leftarrow \text{Chal}(k)$ , is executed by a verifier.  $\text{Chal}$  takes a security parameter  $k$  and outputs a  $k$ -bit string  $\text{chal}$  sampled uniformly at random.
- **Resp**, denoted  $\text{resp} \leftarrow \text{Resp}(\text{chal}, t)$ , is executed by a prover.  $\text{Resp}$  takes a challenge  $\text{chal}$  and state information  $t$  as input, and outputs a response  $\text{resp}$ .
- **Verify**, denoted  $v \leftarrow \text{Verify}(s, (\text{comm}, \text{chal}, \text{resp}))$  is executed by a verifier.  $\text{Verify}$  takes a statement  $s$  and transcript  $(\text{comm}, \text{chal}, \text{resp})$  as input, and outputs a bit  $v$ , which is 1 if the transcript successfully verifies and 0 otherwise. We assume  $\text{Verify}$  is deterministic.

Moreover, the sigma protocol must be complete: there exists a negligible function  $\mu$ , such that for all security parameters  $k$  and statements and witnesses  $(s, w) \in R$ , we have  $\Pr[(\text{comm}, t) \leftarrow \text{Comm}(s, w); \text{chal} \leftarrow_R \{0, 1\}^k; \text{resp} \leftarrow \text{Resp}(\text{chal}, t) : \text{Verify}(s, (\text{comm}, \text{chal}, \text{resp})) = 1] > 1 - \mu(k)$ .

Some sigma protocols ensure *special soundness* and *special honest-verifier zero knowledge*. We will make use of a result by Bernhard et al. that requires these properties, but we will not need the details of those definitions in our proofs, so we omit them here; see Bernhard et al. [20] for a formalization.

**Definition 19.** Let  $(\text{Gen}, \text{Enc}, \text{Dec})$  be a homomorphic asymmetric encryption scheme and  $\Sigma$  be a sigma protocol for a relation  $R$ .

- $\Sigma$  proves correct key construction if

$$((1^k, pk, m), (sk, r)) \in R \Leftrightarrow (pk, sk, m) = \text{Gen}(1^k; r)$$

Further, suppose that  $(pk, sk, m)$  is the output of  $\text{Gen}(1^k; r)$ , for some coins  $r$ .

- $\Sigma$  proves plaintext knowledge in a subspace if

$$\begin{aligned} ((pk, c, m'), (m, r)) &\in R \\ \Leftrightarrow c &= \text{Enc}(pk, m; r) \wedge m \in m' \wedge m' \subseteq m. \end{aligned}$$

- $\Sigma$  proves conjunctive plaintext knowledge if

$$\begin{aligned} ((pk, c_1, \dots, c_k), (m_1, r_1, \dots, m_k, r_k)) &\in R \\ \Leftrightarrow \bigwedge_{1 \leq i \leq k} c_i &= \text{Enc}(pk, m_i; r_i) \wedge m_i \in m. \end{aligned}$$

- $\Sigma$  proves correct reencryption if

$$\begin{aligned} ((pk, \mathbf{c}, c), (i, r)) &\in R \\ \Leftrightarrow c &= \mathbf{c}[i] \otimes \text{Enc}(pk, \mathbf{e}; r) \wedge 1 \leq i \leq |\mathbf{c}| \end{aligned}$$

where  $\mathbf{c}$  is a vector of ciphertexts encrypted under  $pk$ , and where  $\mathbf{e}$  is an identity element of the encryption scheme's message space with respect to  $\odot$ .

- $\Sigma$  is a plaintext equivalence test (PET) if

$$\begin{aligned} ((pk, c, c', i), sk) &\in R \\ \Leftrightarrow ((i = 0 \wedge \text{Dec}(pk, sk, c) &\neq \text{Dec}(pk, sk, c')) \\ \vee (i = 1 \wedge \text{Dec}(pk, sk, c) &= \text{Dec}(pk, sk, c'))) \\ \wedge \text{Dec}(pk, sk, c) &\neq \perp \wedge \text{Dec}(pk, sk, c') \neq \perp. \end{aligned}$$

- $\Sigma$  is a mixnet if

$$\begin{aligned} ((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) &\in R \\ \Leftrightarrow \bigwedge_{1 \leq i \leq |\mathbf{c}|} \mathbf{c}'[\chi(i)] &= \mathbf{c}[i] \otimes \text{Enc}(pk, \mathbf{e}; \mathbf{r}[i]) \\ \wedge |\mathbf{c}| &= |\mathbf{c}'| = |\mathbf{r}| \end{aligned}$$

where  $\mathbf{c}$  and  $\mathbf{c}'$  are both vectors of ciphertexts encrypted under  $pk$ , and  $\chi$  is a permutation on  $\{1, \dots, |\mathbf{c}|\}$ , and  $\mathbf{e}$  is an identity element of the encryption scheme's message space with respect to  $\odot$ .

- $\Sigma$  proves correct decryption if

$$((pk, c, m), sk) \in R \Leftrightarrow m = \text{Dec}(pk, sk, c).$$

### C. Non-interactive proof systems

A proof system is *non-interactive* if a single message is sent from the prover to the verifier.

**Definition 20** (Non-interactive proof system). A non-interactive proof system for a relation  $R$  is a tuple of algorithms (Prove, Verify) such that:

- **Prove**, denoted  $\sigma \leftarrow \text{Prove}(s, w)$ , is executed by a prover to prove  $(s, w) \in R$ .
- **Verify**, denoted  $v \leftarrow \text{Verify}(s, \sigma)$ , is executed by anyone to check the validity of a proof. We assume Verify is deterministic.

Moreover, the system must be complete: there exists a negligible function  $\mu$ , such that for all statement and witnesses  $(s, w) \in R$ , we have  $\Pr[\sigma \leftarrow \text{Prove}(s, w) : \text{Verify}(s, \sigma) = 1] > 1 - \mu(|s|)$ , where  $|s|$  denotes the length of  $s$ . There are various soundness definitions that can be considered for

*non-interactive proof systems*. We will use simulation-sound extractability, defined below.

We can derive non-interactive proof systems from sigma protocols using the *Fiat-Shamir transformation* [52], which replaces the verifier's challenge with a hash of the prover's commitment, concatenated with the prover's statement.

**Definition 21** (Fiat-Shamir transformation [52]). Given a sigma protocol  $\Sigma = (\text{Comm}, \text{Chal}, \text{Resp}, \text{Verify}_\Sigma)$  for relation  $R$  and a hash function  $\mathcal{H}$ , the Fiat-Shamir transformation, denoted  $\text{FS}(\Sigma, \mathcal{H})$ , is the tuple (Prove, Verify) of PPT algorithms, defined as follows:

Prove( $s, w$ ) =

- 1 (comm,  $t$ )  $\leftarrow$  Comm( $s, w$ );
- 2 chal  $\leftarrow$   $\mathcal{H}$ (comm,  $s$ );
- 3 resp  $\leftarrow$  Resp(chal,  $t$ );
- 4 **return** (comm, resp)

Verify( $s, (\text{comm}, \text{resp})$ ) =

- 1 chal  $\leftarrow$   $\mathcal{H}$ (comm,  $s$ );
- 2 **return** Verify $_\Sigma$ ( $s, (\text{comm}, \text{chal}, \text{resp})$ )

It is straightforward to check that FS produces non-interactive proof systems.

Some non-interactive proof systems ensure *zero knowledge*: anything a verifier can derive about a witness can be derived without interaction with a prover—that is, the prover can be *simulated*. We define zero knowledge in the *random oracle model* [9]. A random oracle can be *programmed* or *patched*. We will not need the details of how patching works in our proofs, so we omit them here; see Bernhard et al. [20] for a formalization.

**Definition 22** (Zero knowledge). Suppose that  $\Sigma$  is a sigma protocol for relation  $R$ , that  $\mathcal{H}$  is a random oracle, and that (Prove, Verify) is a non-interactive proof system. Proof system (Prove, Verify) satisfies zero knowledge if there exists a PPT algorithm  $S$  and a negligible function  $\mu$ , such that for all PPT adversaries  $\mathcal{A}$  and all statements and witnesses  $(x, y) \in R$ , we have

$$\Pr[b \leftarrow \mathcal{A}^{\mathcal{H}, \mathcal{P}_1}() : b = 1] - \Pr[b \leftarrow \mathcal{A}^{\mathcal{H}, \mathcal{P}_2}() : b = 1] \leq \mu(|x|)$$

where oracles  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are defined on inputs  $s$  and  $w$  as follows: if  $(s, w) \notin R$ , then both  $\mathcal{P}_1$  and  $\mathcal{P}_2$  output  $\perp$ , otherwise,  $\mathcal{P}_1$  computes  $\sigma \leftarrow \text{Prove}(s, w)$  and outputs  $\sigma$ , and  $\mathcal{P}_2$  computes  $\tau \leftarrow S(s)$  and outputs  $\tau$ . Moreover, algorithm  $S$  can patch random oracle  $\mathcal{H}$ . The algorithm  $S$  for which the above definition holds is called a *simulator* for (Prove, Verify).

Some zero knowledge non-interactive proof systems ensure *simulation sound extractability*: an extractor can recover witnesses from proofs by *rewinding* the prover, as discussed below. We use extractors in our proofs of theorems, below, to obtain witnesses from proofs.

**Definition 23** (Simulation sound extractability [20], [64]). Suppose that  $\Sigma$  is a sigma protocol for relation  $R$ , that  $\mathcal{H}$  is

a random oracle, and that (Prove, Verify) is a non-interactive proof, system such that  $\text{FS}(\Sigma, \mathcal{H}) = (\text{Prove}, \text{Verify})$ . Further suppose  $\mathcal{S}$  is a simulator for (Prove, Verify) and  $\mathcal{H}$  can be patched by  $\mathcal{S}$ . Proof system (Prove, Verify) satisfies simulation sound extractability if there exists a PPT algorithm  $\mathcal{K}$  and a negligible function  $\mu$ , such that for all adversaries  $\mathcal{A}$ , coins  $r$ , and statements and witnesses  $(x, w) \in R$ , we have<sup>31</sup>

$$\begin{aligned} & \Pr[\mathbf{P} \leftarrow (); \mathbf{Q} \leftarrow \mathcal{A}^{\mathcal{H}, \mathcal{P}}(-; r); \mathbf{W} \leftarrow \mathcal{K}^{\mathcal{A}'}(\mathbf{H}, \mathbf{P}, \mathbf{Q}) : \\ & (|\mathbf{Q}| = |\mathbf{W}| \Rightarrow \exists j \in \{1, \dots, |\mathbf{Q}|\} \cdot (\mathbf{Q}[j][1], \mathbf{W}[j]) \notin R) \\ & \wedge \forall (s, \sigma) \in \mathbf{Q}, (t, \tau) \in \mathbf{P} \cdot \text{Verify}(s, \sigma) = 1 \wedge \sigma \neq \tau] \leq \mu(|x|) \end{aligned}$$

where  $\mathcal{A}(-; r)$  denotes running adversary  $\mathcal{A}$  with an empty input and random coins  $r$ , where  $\mathbf{H}$  is a transcript of the random oracle's input and output, and where oracles  $\mathcal{A}'$  and  $\mathcal{P}$  are defined below:

- $\mathcal{A}'()$ . Computes  $\mathbf{Q}' \leftarrow \mathcal{A}(-; r)$ , forwarding any of  $\mathcal{A}'$ 's oracle calls to  $\mathcal{K}$ , and outputs  $\mathbf{Q}'$ . By running  $\mathcal{A}(-; r)$ ,  $\mathcal{K}$  is rewinding the adversary.
- $\mathcal{P}(s)$ . Computes  $\sigma \leftarrow \mathcal{S}(s)$ ;  $\mathbf{P} \leftarrow (\mathbf{P}[1], \dots, \mathbf{P}[|\mathbf{P}|], (s, \sigma))$  and outputs  $\sigma$ .

Algorithm  $\mathcal{K}$  is an extractor for (Prove, Verify).

Our definition of simulation sound extractability in the random oracle model is an analogue of Groth's definition in the common reference string model [64, §2]. (See Bernhard et al. [20, §1] for a detailed comparison.) Our presentation of simulation sound extractability differs from the presentation by Bernhard et al. [20] by formalizing some of the details.

Bernhard et al. [20] show that non-interactive proof systems derived using the Fiat-Shamir transformation satisfy zero knowledge and simulation sound extractability:

**Theorem 9** (from [20]). *Let  $\Sigma$  be a sigma protocol for relation  $R$ , and let  $\mathcal{H}$  be a random oracle. If  $\Sigma$  satisfies special soundness and special honest verifier zero knowledge, then  $\text{FS}(\Sigma, \mathcal{H})$  satisfies simulation sound extractability.*

The Fiat-Shamir transformation can be generalized to include an optional string  $m$  in the hashes produced by functions Prove and Verify. We write  $\text{Prove}(s, w, m)$  and  $\text{Verify}(s, (\text{comm}, \text{resp}), m)$  for invocations of Prove and Verify which include an optional string. When  $m$  is provided, it is included in the hashes in both algorithms. That is, given  $\text{FS}(\Sigma, \mathcal{H}) = (\text{Prove}, \text{Verify})$ , the hashes are computed as follows in both algorithms:  $\text{chal} \leftarrow \mathcal{H}(\text{comm}, s, m)$ . Theorem 9 can be extended to this generalization.

## APPENDIX B VARIANTS OF Exp-IV

Our individual verifiability experiment with external authentication (§II-B1) can be equivalently formulated as an experiment that challenges  $\mathcal{A}$  to predict the output of Vote:

<sup>31</sup>We extend set membership notation to vectors: we write  $x \in \mathbf{x}$  if  $x$  is an element of the set  $\{\mathbf{x}[i] : 1 \leq i \leq |\mathbf{x}|\}$

```
Exp-IV-Ext'(Π, A, k) =
1 (PKT, nC, β, b) ← A(k);
2 b' ← Vote(PKT, nC, β, k);
3 if b = b' ∧ b' ≠ ⊥ then
4 | return 1
5 else
6 | return 0
```

**Proposition 10.** *Given an election scheme  $\Pi$ , we have*

$$\begin{aligned} & \forall \mathcal{A} \exists \mu \forall k \cdot \text{Succ}(\text{Exp-IV-Ext}(\Pi, \mathcal{A}, k)) \leq \mu(k) \\ & \Leftrightarrow \forall \mathcal{A}' \exists \mu' \forall k' \cdot \text{Succ}(\text{Exp-IV-Ext}'(\Pi, \mathcal{A}', k')) \leq \mu'(k'), \end{aligned}$$

where  $\mathcal{A}$  and  $\mathcal{A}'$  are PPT adversaries,  $\mu$  and  $\mu'$  are negligible functions, and  $k$  and  $k'$  are security parameters.

Intuitively, if  $\mathcal{A}$  can predict the output of Vote, then  $\mathcal{A}$  can use that prediction to generate a collision. And if  $\mathcal{A}$  can generate collisions, then  $\mathcal{A}$  can use them to predict outputs.

*Proof.* For the forward implication, suppose  $\mathcal{A}'$  is a PPT adversary such that  $\text{Succ}(\text{Exp-IV-Ext}'(\Pi, \mathcal{A}', k')) > \frac{1}{p(k')}$  for some polynomial  $p$  and security parameter  $k'$ . We construct an adversary  $\mathcal{A}$  against Exp-IV-Ext. On input  $k'$ , adversary  $\mathcal{A}$  computes  $(PK_{\mathcal{T}}, n_C, \beta, b) \leftarrow \mathcal{A}'(k')$  and outputs  $(PK_{\mathcal{T}}, n_C, \beta, \beta)$ . Since  $\mathcal{A}'$  wins Exp-IV-Ext' with non-negligible probability, we have

$$\Pr[b' \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k') : b = b' \wedge b \neq \perp] > \frac{1}{p(k')}.$$

Moreover, since calls to algorithm Vote are independent, we have

$$\begin{aligned} & \Pr[b_1 \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k'); \\ & \quad b_2 \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k') \\ & : b_1 = b \wedge b_2 = b \wedge b_1 \neq \perp \wedge b_2 \neq \perp] > \frac{1}{p(k')^2}. \end{aligned}$$

It follows that  $\text{Succ}(\text{Exp-IV-Ext}(\Pi, \mathcal{A}, k')) > \frac{1}{p(k')^2}$ .

For the reverse implication, suppose  $\mathcal{A}$  is a PPT adversary such that  $\text{Succ}(\text{Exp-IV-Ext}(\Pi, \mathcal{A}, k)) > \frac{1}{p(k)}$  for some polynomial  $p$  and security parameter  $k$ . We construct an adversary  $\mathcal{A}'$  against Exp-IV-Ext'. On input  $k$ , adversary  $\mathcal{A}'$  computes  $(PK_{\mathcal{T}}, n_C, \beta_1, \beta_2) \leftarrow \mathcal{A}(k)$ ;  $b_1 \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta_1, k)$  and outputs  $(PK_{\mathcal{T}}, n_C, \beta_2, b_1)$ . Since  $\mathcal{A}$  wins Exp-IV-Ext with probability no less than  $\frac{1}{p(k)}$ , we have

$$\Pr[b_2 \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta_2, k) : b_1 = b_2 \wedge b_1 \neq \perp] > \frac{1}{p(k)}.$$

It follows that  $\text{Succ}(\text{Exp-IV-Int}'(\Pi, \mathcal{A}', k)) > \frac{1}{p(k)}$ .  $\square$

Our individual verifiability experiment with internal authentication (§IV-B1) can also be reformulated as an experiment that challenges  $\mathcal{A}$  to predict the output of Vote algorithms:

```

Exp-IV-Int'(II, A, k) =
1 (PKT, nV) ← A(k);
2 for 1 ≤ i ≤ nV do (pki, ski) ← Register(PKT, k)
3 L ← {pk1, ..., pknV};
4 Crpt ← ∅;
5 (nC, β, i, b) ← AC(L);
6 b' ← Vote(ski, PKT, nC, β, k);
7 if b = b' ∧ b' ≠ ⊥ ∧ ski ∉ Crpt then
8   | return 1
9 else
10  | return 0

```

Similarly to Section IV-B1, the adversary is given access to oracle  $C$  and the voter index output on line 5 must be legal with respect to  $n_V$ .

Experiment Exp-IV-Int' is strictly stronger than our original experiment Exp-IV-Int, since predicting the output of Vote does not imply the existence of collisions, whereas collisions can be used to predict the output of Vote. For instance, consider the following variant of Nonce (Definition 5):

**Definition 24.** Election scheme Nonce' is defined as follows:

- Setup( $k$ ) outputs  $(\perp, \perp, \infty, \infty)$ .
- Register( $PK_{\mathcal{T}}, k$ ) computes  $r \in \mathbb{Z}_{2^k}$  and outputs  $(r, r)$ .
- Vote( $r, PK_{\mathcal{T}}, n_C, \beta, k$ ) outputs  $(r, \beta)$ .
- 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 the votes on  $BB$  for which the nonce is in  $L$ , and outputs  $(\mathbf{X}, \perp)$ .
- Verify( $PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k$ ) outputs 1 if  $(\mathbf{X}, P) = \text{Tally}(\perp, \perp, BB, L, n_C, k)$  and 0 otherwise.

Intuitively, an adversary can predict the output of Vote, because the algorithm is deterministic and the electoral roll lists private credentials. However, the Register algorithm ensures that voters' credentials are distinct with overwhelming probability, hence, instantiations of the Vote algorithm with distinct voter credentials will never collide.

**Proposition 11.** Given an election scheme  $\Pi$ , PPT adversary  $\mathcal{A}$ , negligible function  $\mu$ , and security parameter  $k$ , if  $\text{Succ}(\text{Exp-IV-Int}'(\Pi, \mathcal{A}, k)) \leq \mu(k)$ , then there exists a PPT adversary  $\mathcal{B}$  such that  $\text{Succ}(\text{Exp-IV-Int}(\Pi, \mathcal{B}, k)) \leq \mu(k)$ .

The proof of Proposition 11 is similar to the reverse implication proof of Proposition 10.

## APPENDIX C HELIOS 4.0 SCHEME

We formalize a generic construction for Helios-like election schemes (Figure 1). Our construction is parameterized on the choice of homomorphic encryption scheme and sigma protocols.

Setup generates the tallier's key pair. The public key includes a non-interactive proof that the key pair is correctly constructed. Vote takes a choice  $\beta \in \{1, \dots, n_C\}$  and outputs ciphertexts  $c_1, \dots, c_{n_C-1}$  such that if  $\beta < n_C$ , then ciphertext  $c_\beta$  contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. Vote

also outputs proofs  $\sigma_1, \dots, \sigma_{n_C}$  so that this can be verified, in particular, proof  $\sigma_j$  demonstrates that the ciphertext  $c_j$  contains 0 or 1 for all  $1 \leq j \leq n_C - 1$ , and the proof  $\sigma_{n_C}$  demonstrates that the homomorphic combination of ciphertexts  $c_1 \otimes \dots \otimes c_{n_C}$  contains 0 or 1 (i.e., the voter's ballot contains a vote for exactly one candidate). Tally homomorphically combines ciphertexts representing votes for a particular candidate and decrypts the homomorphic combinations. The number of votes for a candidate  $\beta \in \{1, \dots, n_C - 1\}$  is simply the homomorphic combination of the ballots for that candidate; the number of votes for candidate  $n_C$  is equal to the number of votes for all other candidates subtracted from the total number of valid ballots on the bulletin board. Verify checks that each of the above steps has been performed correctly.

Lemmata 12–14 demonstrate that generalized Helios is a construction for election schemes.

**Lemma 12.** Helios( $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}$ ) satisfies Correctness, where  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1.

The proof of Lemma 12 is similar to the proof of Proposition 17.

**Lemma 13.** Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1. Further suppose that  $\Sigma_2$  satisfies special soundness and special honest verifier zero-knowledge, and  $\mathcal{H}$  is a random oracle. We have Helios( $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}$ ) satisfies Completeness.

*Proof.* Let Helios( $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}$ ) = (Setup, Vote, Tally, Verify), FS( $\Sigma_1, \mathcal{H}$ ) = (ProveKey, VerKey), FS( $\Sigma_2, \mathcal{H}$ ) = (ProveCiph, VerCiph), and FS( $\Sigma_3, \mathcal{H}$ ) = (ProveDec, VerDec). Suppose  $k$  is a security parameter,  $BB$  is a bulletin board, and  $n_C$  is an integer. Further suppose  $(PK_{\mathcal{T}}, sk)$  is a key pair,  $m_B$  and  $m_C$  are integers, and  $(\mathbf{X}, P)$  is a tally, such that  $(PK_{\mathcal{T}}, sk, m_B, m_C) \leftarrow \text{Setup}(k)$  and  $(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, sk, BB, n_C, k)$ . Moreover, suppose  $|BB| \leq m_B$ . We focus on the case  $n_C > 1$ ; the case  $n_C = 1$  is similar. By definition of Setup, there exist coins  $s$  such that  $(pk, sk, m) = \text{Gen}(1^k; s)$ ,  $PK_{\mathcal{T}} \leftarrow (pk, m, \rho)$  and  $m_B$  is the largest integer such that  $\{0, \dots, m_B\} \subseteq m$ , where  $\rho$  is an output of ProveKey( $(1^k, pk, m), (sk, s)$ ). By definition of Tally, we have  $\mathbf{X}$  is a vector of length  $n_C$  and  $P$  is a vector of length  $n_C - 1$ . It follows that Verify can successfully parse  $\mathbf{X}, P$ , and  $PK_{\mathcal{T}}$ . Moreover, by the completeness of (ProveKey, VerKey), we have VerKey( $(1^k, pk, m), \rho$ ) = 1 with overwhelming probability. Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  satisfying the conditions given by the tally algorithm. If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then  $\mathbf{X}$  is a zero-filled vector and Verify outputs 1, concluding our proof, otherwise, we proceed as follows. Since  $\{b_1, \dots, b_\ell\}$  is a subset of  $BB$ , we have  $\ell \leq m_B$ . By definition of Tally, we have for all  $1 \leq i \leq \ell$  that  $\bigwedge_{j=1}^{n_C-1} \text{VerCiph}((pk, b_i[j], \{0, 1\}), b_i[j + n_C - 1], j) = 1$ . By Theorem 9, we have (ProveCiph, VerCiph) satisfies simulation sound extractability, hence, for all  $1 \leq i \leq \ell$  and all  $1 \leq j \leq n_C - 1$  we have  $b_i[j]$  is a ciphertext with overwhelming probability. It

**Fig. 1** Generalized Helios

Suppose  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$  is an additively homomorphic asymmetric encryption scheme with a message space that, for sufficiently large security parameters, includes  $\{0, 1\}$ ,  $\Sigma_1$  proves correct key construction,  $\Sigma_2$  proves plaintext knowledge in a subspace,  $\Sigma_3$  proves correct decryption, and  $\mathcal{H}$  is a hash function. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ ,  $\text{FS}(\Sigma_2, \mathcal{H}) = (\text{ProveCiph}, \text{VerCiph})$ , and  $\text{FS}(\Sigma_3, \mathcal{H}) = (\text{ProveDec}, \text{VerDec})$ . We define *generalized Helios*  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$  as follows.

- **Setup**( $k$ ). Select coins  $s$ , compute  $(pk, sk, m) \leftarrow \text{Gen}(1^k; s); \rho \leftarrow \text{ProveKey}((1^k, pk, m), (sk, s)); PK_{\mathcal{T}} \leftarrow (pk, m, \rho)$ , let  $m_B$  be the largest integer such that  $\{0, \dots, m_B\} \subseteq m$ , and output  $(PK_{\mathcal{T}}, sk, m_B, \infty)$ .
- **Vote**( $PK_{\mathcal{T}}, n_C, \beta, k$ ). Parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output  $\perp$  if parsing fails or  $\text{VerKey}((1^k, pk, m), \rho) \neq 1 \vee \beta \notin \{1, \dots, n_C\}$ . Select coins  $r_1, \dots, r_{n_C-1}$  and compute:

```

for  $1 \leq j \leq n_C - 1$  do
  if  $j = \beta$  then  $m_j \leftarrow 1$  else  $m_j \leftarrow 0$ 
   $c_j \leftarrow \text{Enc}(pk, m_j; r_j)$ ;
   $\sigma_j \leftarrow \text{ProveCiph}((pk, c_j, \{0, 1\}), (m_j, r_j), j)$ 
 $c \leftarrow c_1 \otimes \dots \otimes c_{n_C-1}$ ;
 $m \leftarrow m_1 \odot \dots \odot m_{n_C-1}$ ;
 $r \leftarrow r_1 \oplus \dots \oplus r_{n_C-1}$ ;
 $\sigma_{n_C} \leftarrow \text{ProveCiph}((pk, c, \{0, 1\}), (m, r), n_C)$ 

```

Output ballot  $(c_1, \dots, c_{n_C-1}, \sigma_1, \dots, \sigma_{n_C})$ .

- **Tally**( $PK_{\mathcal{T}}, sk, BB, n_C, k$ ). Initialize vectors  $\mathbf{X}$  of length  $n_C$  and  $\mathbf{P}$  of length  $n_C - 1$ . Compute **for**  $1 \leq j \leq n_C$  **do**  $\mathbf{X}[j] \leftarrow 0$ . Parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output  $(\mathbf{X}, \mathbf{P})$  if parsing fails. Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a vector of length  $2 \cdot n_C - 1$  and  $\bigwedge_{j=1}^{n_C-1} \text{VerCiph}((pk, b_i[j], \{0, 1\}), b_i[j + n_C - 1], j) = 1 \wedge \text{VerCiph}((pk, b_i[1] \otimes \dots \otimes b_i[n_C - 1], \{0, 1\}), b_i[2 \cdot n_C - 1], n_C) = 1$ . If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then output  $(\mathbf{X}, \mathbf{P})$ , otherwise, compute:

```

for  $1 \leq j \leq n_C - 1$  do
   $c \leftarrow b_1[j] \otimes \dots \otimes b_\ell[j]$ ;
   $\mathbf{X}[j] \leftarrow \text{Dec}(pk, sk, c)$ ;
   $\mathbf{P}[j] \leftarrow \text{ProveDec}((pk, c, \mathbf{X}[j]), sk)$ 
 $\mathbf{X}[n_C] \leftarrow \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$ ;

```

Output  $(\mathbf{X}, \mathbf{P})$ .

- **Verify**( $PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, \mathbf{P}, k$ ). Parse  $\mathbf{X}$  as a vector of length  $n_C$ , parse  $\mathbf{P}$  as a vector of length  $n_C - 1$ , parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output 0 if parsing fails or  $\text{VerKey}((1^k, pk, m), \rho) \neq 1$ . Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  satisfying the conditions given by the tally algorithm and let  $m_B$  be the largest integer such that  $\{0, \dots, m_B\} \subseteq m$ . If  $\{b_1, \dots, b_\ell\} = \emptyset \wedge \bigwedge_{j=1}^{n_C} \mathbf{X}[j] = 0$  or  $\bigwedge_{j=1}^{n_C-1} \text{VerDec}((pk, b_1[j] \otimes \dots \otimes b_\ell[j], \mathbf{X}[j]), \mathbf{P}[j]) = 1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j] \wedge 1 \leq \ell \leq m_B$ , then output 1, otherwise, output 0.

The above algorithms assume  $n_C > 1$  and we define special cases of Vote, Tally and Verify when  $n_C = 1$ :

- **Vote**( $PK_{\mathcal{T}}, n_C, \beta, k$ ). Parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output  $\perp$  if parsing fails or  $\text{VerKey}((1^k, pk, m), \rho) \neq 1 \vee \beta \neq 1$ . Select coins  $r$ , compute  $m \leftarrow 1; c \leftarrow \text{Enc}(pk, m; r); \sigma \leftarrow \text{ProveCiph}((pk, c, \{0, 1\}), (m, r))$ , and output ballot  $(c, \sigma)$ .
- **Tally**( $PK_{\mathcal{T}}, sk, BB, n_C, k$ ). Initialize  $\mathbf{X}$  and  $\mathbf{P}$  as vectors of length 1. Compute  $\mathbf{X}[1] \leftarrow 0$ . Parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output  $(\mathbf{X}, \mathbf{P})$  if parsing fails. Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a vector of length 2 and  $\text{VerCiph}((pk, b_i[1], \{0, 1\}), b_i[2]) = 1$ . If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then output  $(\mathbf{X}, \mathbf{P})$ . Otherwise, compute  $c \leftarrow b_1[1] \otimes \dots \otimes b_\ell[1]; \mathbf{X}[1] \leftarrow \text{Dec}(pk, sk, c); \mathbf{P}[1] \leftarrow \text{ProveDec}((pk, c, \mathbf{X}[1]), sk)$  and output  $(\mathbf{X}, \mathbf{P})$ .
- **Verify**( $PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, \mathbf{P}, k$ ). Parse  $\mathbf{X}$  and  $\mathbf{P}$  as vectors of length 1, and parse  $PK_{\mathcal{T}}$  as a vector  $(pk, m, \rho)$ . Output 0 if parsing fails or  $\text{VerKey}((1^k, pk, m), \rho) \neq 1$ . Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  satisfying the conditions given by the tally algorithm and let  $m_B$  be the largest integer such that  $\{0, \dots, m_B\} \subseteq m$ . If  $\{b_1, \dots, b_\ell\} = \emptyset \wedge \mathbf{X}[1] = 0$  or  $\text{VerDec}((pk, b_1[1] \otimes \dots \otimes b_\ell[1], \mathbf{X}[1]), \mathbf{P}[1]) = 1 \wedge 1 \leq \ell \leq m_B$ , then output 1, otherwise, output 0.

follows for all  $1 \leq j \leq n_C - 1$  that  $b_1[j] \otimes \dots \otimes b_\ell[j]$  is a ciphertext with overwhelming probability. By definition of Tally and the completeness of  $(\text{ProveDec}, \text{VerDec})$ , we have  $\bigwedge_{j=1}^{n_C-1} \text{VerDec}((pk, b_1[j] \otimes \dots \otimes b_\ell[j], \mathbf{X}[j]), \mathbf{P}[j]) =$

$1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$  with overwhelming probability, hence, Verify outputs 1 with overwhelming probability, concluding our proof.  $\square$

**Definition 25** (Collision-free). *Suppose*  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$

is an asymmetric encryption scheme,  $\Sigma_1$  proves correct key construction,  $\mathcal{H}$  is a hash function, and  $\mathfrak{m}$  is a message space. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . If for all security parameters  $k$ , public keys  $pk$ , proofs  $\rho$ , messages  $m_1, m_2 \in \mathfrak{m}$ , and coins  $r_1$  and  $r_2$ , we have

$$\begin{aligned} \text{VerKey}((1^k, pk, \mathfrak{m}), \rho) = 1 \wedge (m_1 \neq m_2 \vee r_1 \neq r_2) \\ \Rightarrow \text{Enc}(pk, m_1; r_1) \neq \text{Enc}(pk, m_2; r_2) \end{aligned}$$

Then we say  $\Gamma$  is collision-free for  $\mathfrak{m}$ .

**Lemma 14.** *Suppose  $\Gamma$ ,  $\Sigma_1$ ,  $\Sigma_2$ ,  $\Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1. Further suppose  $\Gamma$  is collision-free for  $\{0, 1\}$ . We have  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$  satisfies Injectivity.*

*Proof.* Let  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ ,  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$ , and  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . Suppose  $k$  is a security parameter,  $PK_{\mathcal{T}}$  is a public key,  $n_C$  is an integer, and  $\beta$  and  $\beta'$  are choices such that  $\beta \neq \beta'$ . Further suppose  $b$  and  $b'$  are ballots such that  $b \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$ ,  $b' \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$ ,  $b \neq \perp$ , and  $b' \neq \perp$ . By definition of  $\text{Vote}$ , we have  $PK_{\mathcal{T}}$  is a vector  $(pk, \mathfrak{m}, \rho)$  and  $\text{VerKey}((1^k, pk, \mathfrak{m}), \rho) = 1$ . Moreover, there exist coins  $r$  and  $r'$  such that

$$b[1] = \text{Enc}(pk, m; r), \text{ where } m = \begin{cases} 1 & \text{if } \beta = 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$b'[1] = \text{Enc}(pk, m'; r'), \text{ where } m' = \begin{cases} 1 & \text{if } \beta' = 1 \\ 0 & \text{otherwise} \end{cases}$$

Since  $\beta \neq \beta'$ , we have  $m \neq m'$ . Furthermore, since  $\Gamma$  is collision-free for  $\{0, 1\}$ , we have  $b[1] \neq b'[1]$  and, therefore,  $b \neq b'$ .  $\square$

Generalized Helios can be instantiated to derive Helios 4.0:

**Definition 26** (Helios 4.0). *Helios 4.0 is  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ , where  $\Gamma$  is additively homomorphic El Gamal [43, §2],  $\Sigma_1$  is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [29, Protocol 2],  $\Sigma_2$  is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [42, Figure 1],  $\Sigma_3$  is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum and Pedersen [30, §3.2], and  $\mathcal{H}$  is a random oracle.*

Although Helios actually uses SHA-256 [98], we assume that  $\mathcal{H}$  is a random oracle to prove Theorem 2. Moreover, we assume the sigma protocols used by Helios 4.0 satisfy the preconditions of generalized Helios—that is, [29, Protocol 2] is a sigma protocol for proving correct key construction, [42, Figure 1] is a sigma protocol for proving plaintext knowledge in a subspace, and [30, §3.2] is a sigma protocol for proving decryption. We leave formally proving this assumption as future work.

To show that Helios 4.0 is an election scheme, we must demonstrate that Correctness, Completeness and Injectivity are

satisfied. Correctness follows immediately from Lemma 12. And we show that Completeness and Injectivity are also satisfied.

First, Completeness. Bernhard et al. [20, §4] remark that the sigma protocol used by Helios 4.0 to prove plaintext knowledge in a subspace satisfies special soundness and special honest verifier zero knowledge, hence, Helios 4.0 satisfies Completeness by Lemma 13.

Secondly, Injectivity. A non-interactive proof system  $(\text{ProveKey}, \text{VerKey})$  derived from a sigma protocol for proving correct key construction is sufficient to ensure that El Gamal is collision-free, assuming algorithm  $\text{VerKey}$  guarantees that public keys are constructed from suitable parameters: if  $\text{VerKey}((1^k, pk, \{0, 1\}), \rho) = 1$ , then there exists  $p, q, g$  and  $h$  such that  $pk = (p, q, g, h)$  and  $(p, q, g)$  are cryptographic parameters—i.e.,  $p = 2 \cdot q + 1$ ,  $|q| = k$ , and  $g$  is a generator of  $\mathbb{Z}_p^*$  of order  $q$ .

**Lemma 15.** *Suppose  $\Sigma_1$  is a sigma protocol that proves correct key construction and  $\mathcal{H}$  is a hash function. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . Further suppose for all security parameters  $k$ , public keys  $pk$ , and proofs  $\rho$ , we have  $\text{VerKey}((1^k, pk, \{0, 1\}), \rho) = 1$  implies  $h \neq 0$  and there exists  $p, q, g$  and  $h$  such that  $pk = (p, q, g, h)$  and  $(p, q, g)$  are cryptographic parameters. It follows that additively homomorphic El Gamal is collision-free for  $\{0, 1\}$ .*

*Proof.* Suppose  $k$  is a security parameter,  $pk$  is a public key,  $\rho$  is a proof,  $m_1, m_2 \in \{0, 1\}$  are messages and  $r_1$  and  $r_2$  are coins such that  $\text{VerKey}((1^k, pk, \{0, 1\}), \rho) = 1$ ,  $m_1 \neq m_2 \vee r_1 \neq r_2$ ,  $pk = (p, q, g, h)$  and  $(p, q, g)$  are cryptographic parameters, for some  $p, q, g$  and  $h$ . Further suppose that  $c_1$  and  $c_2$  are ciphertexts such that  $c_1 = \text{Enc}(pk, m_1; r_1)$ ,  $c_2 = \text{Enc}(pk, m_2; r_2)$ , and  $\text{Enc}$  is El Gamal's encryption algorithm. If  $r_1 \neq r_2$ , then we proceed as follows. By definition of  $\text{Enc}$ , we have  $c_1[1] = g^{r_1} \pmod{p}$  and  $c_2[1] = g^{r_2} \pmod{p}$ . Since  $r_1$  and  $r_2$  are distinct, we have  $g^{r_1} \not\equiv g^{r_2} \pmod{p}$ . (We implicitly assume that coins  $r_1$  and  $r_2$  are selected from the coin space  $\mathbb{Z}_q^*$ , hence,  $g^{r_1} = g^{r_1} \pmod{p}$  and  $g^{r_2} = g^{r_2} \pmod{p}$ .) It follows that  $c_1 \neq c_2$ . Otherwise ( $r_1 = r_2$ ), we have  $m_1 \neq m_2$  and we proceed as follows. By definition of  $\text{Enc}$ , we have  $c_1[2] = h^{r_1} \cdot g_1^{m_1} \pmod{p}$  and  $c_2[2] = h^{r_2} \cdot g_2^{m_2} \pmod{p}$ . Since  $(p, q, g)$  are cryptographic parameters and  $h \neq 0$ , we have  $h^{r_1} \not\equiv h^{r_1} \cdot g \pmod{p}$ , which is sufficient to conclude, because  $m_1, m_2 \in \{0, 1\}$ .  $\square$

The sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [29, Protocol 2] does not explicitly require the suitability of cryptographic parameters to be checked, hence, Lemma 15 is not immediately applicable. Nonetheless, we can trivially make the necessary checks explicit and, hence, the non-interactive proof system derived from the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. is sufficient to ensure that El Gamal is collision-free. It follows that Helios 4.0 satisfies Injectivity, hence, Helios 4.0 is an election scheme.

APPENDIX D

PROOF: HELIOS 4.0 IS VERIFIABLE

Elections schemes constructed from generalized Helios satisfy individual (§D-A) and universal (§D-B) verifiability, hence, such schemes satisfy election verifiability with external authentication (§D-C). It follows that Helios 4.0 satisfies election verifiability (§D-D).

A. Individual verifiability

**Proposition 16.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1. Further suppose that  $\Gamma$  is collision-free for  $\{0, 1\}$ . We have  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$  satisfies individual verifiability.*

The proof of Proposition 16 is similar to the proof of Lemma 14.

*Proof.* Let  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$  and  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . Suppose  $k$  is a security parameter,  $PK_{\mathcal{T}}$  is a public key,  $n_C$  is an integer, and  $\beta$  and  $\beta'$  are choices. Further suppose that  $b$  and  $b'$  are ballots such that  $b \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$ ,  $b' \leftarrow \text{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$ ,  $b \neq \perp$ , and  $b' \neq \perp$ . By definition of  $\text{Vote}$ , we have  $PK_{\mathcal{T}}$  parses as a vector  $(pk, m, \rho)$  and  $\text{VerKey}((1^k, pk, m), \rho) = 1$ . Moreover,  $b[1]$  and  $b'[1]$  are ciphertexts such that  $b[1] \leftarrow \text{Enc}(pk, m)$  and  $b'[1] \leftarrow \text{Enc}(pk, m')$ , where  $m, m' \in \{0, 1\}$ . Furthermore, the ciphertexts are constructed using random coins—i.e., the coins used by  $b[1]$  and  $b'[1]$  will be distinct with overwhelming probability. Since  $\Gamma$  is collision-free for  $\{0, 1\}$ , we have  $b[1] \neq b'[1]$  and  $b \neq b'$  with overwhelming probability, concluding our proof.  $\square$

B. Universal verifiability

**Proposition 17.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1. Further suppose that  $\Sigma_1, \Sigma_2$  and  $\Sigma_3$  satisfy special soundness and special honest verifier zero-knowledge, and  $\mathcal{H}$  is a random oracle. We have  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$  satisfies universal verifiability.*

*Proof.* Let  $\Pi = \text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ ,  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ ,  $\text{FS}(\Sigma_2, \mathcal{H}) = (\text{ProveCiph}, \text{VerCiph})$ , and  $\text{FS}(\Sigma_3, \mathcal{H}) = (\text{ProveDec}, \text{VerDec})$ . By Theorem 9, each of the non-interactive proof systems satisfies simulation sound extractability.

Suppose  $k$  is a security parameter and  $\mathcal{A}$  is a PPT adversary. Further suppose that an execution of  $\text{Exp-UV-Ext}(\Pi, \mathcal{A}, k)$  computes

$$\begin{aligned} (PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) &\leftarrow \mathcal{A}(k); \\ \mathbf{Y} &\leftarrow \text{correct-tally}(PK_{\mathcal{T}}, BB, n_C, k) \end{aligned}$$

such that  $\text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$ . (If  $\text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) \neq 1$ , then we can conclude immediately.) We focus on the case  $n_C > 1$ ; the case  $n_C = 1$  is similar.

By definition of the verification algorithm, vector  $\mathbf{X}$  is of length  $n_C$  and  $P$  is a vector of length  $n_C - 1$ . Moreover,  $PK_{\mathcal{T}}$

is a vector  $(pk, m, \rho)$ . Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a vector of length  $2 \cdot n_C - 1$  and  $\bigwedge_{j=1}^{n_C-1} \text{VerCiph}((pk, b_i[j], \{0, 1\}), b_i[j + n_C - 1], j) = 1 \wedge \text{VerCiph}((pk, b_i[1] \otimes \dots \otimes b_i[n_C - 1], \{0, 1\}), b_i[2 \cdot n_C - 1], n_C) = 1$ .

We have for all choices  $\beta \in \{1, \dots, n_C\}$ , coins  $r$  and ballots  $b = \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r)$  that  $b \notin BB \setminus \{b_1, \dots, b_\ell\}$  with overwhelming probability, since such an occurrence would imply a contradiction:  $\{b_1, \dots, b_\ell\}$  is not the largest subset of  $BB$  satisfying the conditions given by the tally algorithm, because  $b$  is a vector of length  $2 \cdot n_C - 1$  such that  $\bigwedge_{j=1}^{n_C-1} \text{VerCiph}((pk, b[j], \{0, 1\}), b[j + n_C - 1], j) = 1 \wedge \text{VerCiph}((pk, b[1] \otimes \dots \otimes b[n_C - 1], \{0, 1\}), b[2 \cdot n_C - 1], n_C) = 1$  with overwhelming probability, but  $b \notin \{b_1, \dots, b_\ell\}$ . It follows that:

$$\begin{aligned} \text{correct-tally}(PK_{\mathcal{T}}, BB, n_C, k) \\ = \text{correct-tally}(PK_{\mathcal{T}}, \{b_1, \dots, b_\ell\}, n_C, k) \end{aligned} \quad (1)$$

A proof of (1) follows from the definition of function  $\text{correct-tally}$ .

We proceed by distinguishing two cases.

**Case I:**  $\{b_1, \dots, b_\ell\} = \emptyset$ . By definition of function  $\text{correct-tally}$  and (1), we have  $\mathbf{Y}$  is a vector of length  $n_C$  such that  $\bigwedge_{j=1}^{n_C} \mathbf{Y}[j] = 0$ . Since  $\bigwedge_{i=j}^{n_C} \mathbf{X}[j] = 0$ , we have  $\mathbf{X} = \mathbf{Y}$  by definition of the verification algorithm.

**Case II:**  $\{b_1, \dots, b_\ell\} \neq \emptyset$ . By definition of the verification algorithm, we have  $\text{VerKey}((1^k, pk, m), \rho) = 1$ . Moreover, by simulation sound extractability, we are assured that  $pk$  is an output of  $\text{Gen}$  with overwhelming probability—i.e., there exists  $s$  and  $sk$  such that  $(pk, sk, m) = \text{Gen}(1^k; s)$ .

By simulation sound extractability, with overwhelming probability, for all  $1 \leq i \leq \ell$  there exists messages  $m_{i,1}, \dots, m_{i,n_C-1} \in \{0, 1\}$  and coins  $r_{i,1}, \dots, r_{i,2 \cdot n_C - 2}$  such that for all  $1 \leq j \leq n_C - 1$  we have

$$\begin{aligned} b_i[j + n_C - 1] &= \text{ProveCiph}((pk, b_i[j], \{0, 1\}), \\ &\quad (m_{i,j}, r_{i,j}), j; r_{i,j+n_C-1}) \end{aligned}$$

and

$$b_i[j] = \text{Enc}(pk, m_{i,j}; r_{i,j}).$$

Moreover, for all  $1 \leq i \leq \ell$  we have  $\sum_{j=1}^{n_C-1} m_{i,j} \in \{0, 1\}$  and there exist coins  $r_{i,2 \cdot n_C - 1}$  such that

$$\begin{aligned} b_i[2 \cdot n_C - 1] &= \text{ProveCiph}(pk, c, \{0, 1\}), \\ &\quad (m, r), n_C; r_{i,2 \cdot n_C - 1}) \end{aligned}$$

with overwhelming probability, where  $c \leftarrow b_i[1] \otimes \dots \otimes b_i[n_C - 1]$ ,  $m \leftarrow m_{i,1} \odot \dots \odot m_{i,n_C-1}$ , and  $r \leftarrow r_{i,1} \oplus \dots \oplus r_{i,n_C-1}$ .

By inspection of  $\text{Vote}$ , for all  $1 \leq i \leq \ell$  there exists  $\beta_i, r_i$  such that

$$b_i = \text{Vote}(PK_{\mathcal{T}}, n_C, \beta_i, k; r_i)$$

and either  $\beta_i = n_C \wedge \bigwedge_{j=1}^{n_C-1} m_{i,j} = 0$  or  $\beta_i \in \{1, \dots, n_C - 1\} \wedge m_{i,\beta_i} = 1 \wedge \bigwedge_{j \in \{1, \dots, \beta_i - 1, \beta_i + 1, \dots, n_C - 1\}} m_{i,j} = 0$ . It follows for all  $1 \leq i \leq \ell$  and  $1 \leq j \leq n_C - 1$  that:

$$m_{i,j} = 0 \iff \beta_i = n_C \vee \beta_i \neq j \quad (2)$$

$$m_{i,j} = 1 \iff \beta_i = j \quad (3)$$

Moreover, for all  $1 \leq i \leq \ell$  we have:

$$\sum_{j=1}^{n_C-1} m_{i,j} = 0 \iff \beta_i = n_C \quad (4)$$

Furthermore, we have the following facts:

**Fact 1.** For all integers  $\beta$  and  $k$  such that  $1 \leq \beta \leq n_C$ , we have:

$$\begin{aligned} \exists^{=k} b \in (\{b_1, \dots, b_\ell\} \setminus \{\perp\}) : \\ \exists r : b = \text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r) \\ \iff \exists^{=k} i \in \{1, \dots, \ell\} : \beta = \beta_i \end{aligned}$$

**Fact 2.** For all integers  $j$  and  $k$  such that  $1 \leq j \leq n_C - 1$ , we have:

$$\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = j \iff k = \sum_{i=1}^{\ell} m_{i,j}$$

*Proof of Fact 2.* For the forward implication, suppose  $j, k$  are integers such that  $1 \leq j \leq n_C - 1$  and  $\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = j$ . We proceed by induction on  $\ell$ . In the base case ( $\ell = 0$ ), we have  $k = 0$ , hence,  $k = \sum_{i=1}^{\ell} m_{i,j}$ . In the inductive case, we distinguish two cases. Case I:  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = j$  holds. We have  $\beta_\ell \neq j$  by definition of the counting quantifier and, hence,  $m_{i,j} = 0$  by (2). By our induction hypothesis, we derive  $k = \sum_{i=1}^{\ell-1} m_{i,j} = \sum_{i=1}^{\ell} m_{i,j}$ . Case II:  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = j$  does not hold. We have  $\beta_\ell = j$  by definition of the counting quantifier and, hence,  $m_{i,j} = 1$  by (3). Moreover, we have  $\exists^{=k-1} i \in \{1, \dots, \ell - 1\} : \beta_i = j$  holds. By our induction hypothesis, we derive  $k - 1 = \sum_{i=1}^{\ell-1} m_{i,j}$ , that is,  $k = \sum_{i=1}^{\ell} m_{i,j}$ .

For the reverse implication, suppose  $j, k$  are integers such that  $1 \leq j \leq n_C - 1$  and  $k = \sum_{i=1}^{\ell} m_{i,j}$ . We proceed by induction on  $\ell$ . In the base case ( $\ell = 0$ ), we have  $k = 0$ , hence,  $\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = j$ . In the inductive case, we distinguish two cases. Case I:  $k = \sum_{i=1}^{\ell-1} m_{i,j}$ . We have  $m_{\ell,j} = 0$ , hence,  $\beta_\ell \neq j$  by (2). By our induction hypothesis, we have  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = j$ . Since  $\beta_\ell \neq j$ , the result follows. Case II:  $k \neq \sum_{i=1}^{\ell-1} m_{i,j}$ . Since  $m_{\ell,j} \in \{0, 1\}$ , we have  $m_{\ell,j} = 1$ , hence,  $\beta_\ell = j$  by (3). Moreover, we have  $k - 1 = \sum_{i=1}^{\ell-1} m_{i,j}$ . By our induction hypothesis, we derive  $\exists^{=k-1} i \in \{1, \dots, \ell - 1\} : \beta_i = j$ . The result follows.

**Fact 3.** For all integers  $k$ , we have

$$\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = n_C \iff k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$$

*Proof of Fact 3.* For the forward implication, suppose  $\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = n_C$ . We proceed by induction on  $\ell$ . In the base case ( $\ell = 0$ ), we have  $k = 0$ , hence,  $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$ . In the inductive case, we distinguish two cases. Case I:  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = n_C$  holds. We have  $\beta_\ell \neq n_C$  by definition of

the counting quantifier and we derive  $\sum_{j=1}^{n_C-1} m_{\ell,j} \neq 0$  by (4). Moreover, since  $\sum_{j=1}^{n_C-1} m_{\ell,j} \in \{0, 1\}$ , we have  $\sum_{j=1}^{n_C-1} m_{\ell,j} = 1$ . By our induction hypothesis, we derive  $k = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j} = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$ . Case II:  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = n_C$  does not hold. We have  $\beta_\ell = n_C$  by definition of the counting quantifier and we derive  $\sum_{j=1}^{n_C-1} m_{i,j} = 0$  by (4). Moreover, we have  $\exists^{=k-1} i \in \{1, \dots, \ell - 1\} : \beta_i = n_C$  holds. By our induction hypothesis, we derive  $k - 1 = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$ , that is,  $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j} = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$ .

For the reverse implication, suppose  $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$ . We proceed by induction on  $\ell$ . In the base case ( $\ell = 0$ ), we have  $k = 0$ , hence,  $\exists^{=k} i \in \{1, \dots, \ell\} : \beta_i = n_C$ . In the inductive case, we distinguish two cases. Case I:  $k = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$ . We have  $\sum_{j=1}^{n_C-1} m_{\ell,j} = 1$ . Since  $m_{\ell,1}, \dots, m_{\ell,n_C-1} \in \{0, 1\}$ , there exists  $j$  such that  $1 \leq j \leq n_C - 1$  and  $m_{\ell,j} = 1$ , moreover,  $\beta_\ell = j$  by (3), hence,  $\beta_\ell \neq n_C$ . By our induction hypothesis, we derive  $\exists^{=k} i \in \{1, \dots, \ell - 1\} : \beta_i = n_C$ . The result follows. Case II:  $k \neq \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$ . Since  $\sum_{j=1}^{n_C-1} m_{\ell,j} \in \{0, 1\}$ , we have  $\sum_{j=1}^{n_C-1} m_{\ell,j} = 0$ , and we derive  $\beta_i = n_C$  by (4). Moreover, we have  $k - 1 = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$ . By our induction hypothesis, we derive  $\exists^{=k-1} i \in \{1, \dots, \ell - 1\} : \beta_i = n_C$ . The result follows.

We proceed the proof of Proposition 17 using the above facts.

By definition of the verification algorithm, we have  $\bigwedge_{j=1}^{n_C-1} \text{VerDec}((pk, b_1[j] \otimes \dots \otimes b_\ell[j], \mathbf{X}[j]), P[j]) = 1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$ . By simulation sound extractability, we have for all  $1 \leq j \leq n_C - 1$  that  $\mathbf{X}[j] = \text{Dec}(pk, sk, b_1[j] \otimes \dots \otimes b_\ell[j])$  with overwhelming probability, hence,  $\mathbf{X}[j] = m_{1,j} \odot \dots \odot m_{\ell,j}$ , with overwhelming probability. Let  $m_B$  be the largest integer such that  $\{0, \dots, m_B\} \subseteq \mathfrak{m}$ . By definition of the verification algorithm, we have  $\ell \leq m_B$ . It follows that  $m_{1,j} \odot \dots \odot m_{\ell,j} = \sum_{i=1}^{\ell} m_{i,j}$ , hence,

$$\mathbf{X}[j] = \sum_{i=1}^{\ell} m_{i,j}$$

with overwhelming probability. By definition of function *correct-tally*, (1) and Fact 1, we have  $\mathbf{Y}$  is a vector of length  $n_C$  such that for all  $1 \leq \beta \leq n_C$  we have

$$\mathbf{Y}[\beta] = k \text{ if } \exists^{=k} i \in \{1, \dots, \ell\} : \beta = \beta_i$$

It follows by Facts 2 and 3 that for all  $1 \leq \beta \leq n_C$  we have  $\mathbf{X}[\beta] = \mathbf{Y}[\beta]$  with overwhelming probability, hence,  $\mathbf{X} = \mathbf{Y}$  with overwhelming probability.

We have  $\mathbf{X} = \mathbf{Y}$  with overwhelming probability in both cases—i.e.,  $\text{Exp-UV-Ext}(\Pi, \mathcal{A}, k)$  outputs 0 with overwhelming probability and  $\text{Succ}(\text{Exp-UV-Ext}(\Pi, \mathcal{A}, k))$  is negligible, concluding our proof.  $\square$

### C. Election verifiability

By Propositions 16 & 17, election schemes constructed from generalized Helios satisfy election verifiability with external authentication:

**Corollary 18.** *Suppose  $\Gamma$ ,  $\Sigma_1$ ,  $\Sigma_2$ ,  $\Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Figure 1. Further suppose that  $\Gamma$  is collision-free for  $\{0, 1\}$ ,  $\Sigma_1$ ,  $\Sigma_2$  and  $\Sigma_3$  satisfy special soundness and special honest verifier zero-knowledge, and  $\mathcal{H}$  is a random oracle. We have  $\text{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$  satisfies election verifiability with external authentication.*

### D. Proof: Theorem 2

Our proof of Theorem 2 is reliant on Corollary 18. We have already shown that the sigma protocol used by Helios 4.0 to prove discrete logarithms is sufficient to ensure that El Gamal is collision-free (Lemma 15), hence, it remains to show that the sigma protocols used by Helios 4.0 satisfy special soundness and special honest verifier zero-knowledge.

Bernhard et al. [20, §4] remark that the sigma protocols used by Helios 4.0 to prove discrete logarithms and equality between discrete logarithms both satisfy special soundness and special honest verifier zero knowledge, hence, Theorem 9 is applicable. Bernhard et al. also remark that the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms satisfies special soundness and “almost special honest verifier zero knowledge” and argue that “we could fix this[, but] it is easy to see that ... all relevant theorems [including Theorem 9] still hold.” We adopt the same and assume that Theorem 9 is applicable.

*Proof of Theorem 2.* The proof follows from Corollary 18, subject to the applicability of Theorem 9 to the sigma protocol used by Helios 4.0 to prove knowledge of disjunctive equality between discrete logarithms.  $\square$

## APPENDIX E

### PROOF: HELIOS 2.0 IS NOT VERIFIABLE

Bernhard et al. [20] demonstrate that Helios 2.0 [5] is not verifiable and we show that Helios 2.0 does not satisfy Ver-Ext.

**Definition 27** (Weak Fiat-Shamir transformation [20]). *The weak Fiat-Shamir transformation is a function  $wFS$  that is identical to FS, except that it excludes statement  $s$  in the hashes computed by Prove and Verify, as follows:  $\text{chal} \leftarrow \mathcal{H}(\text{comm})$ .*

**Definition 28** (Helios 2.0). *Let  $\widehat{\text{Helios}}$  be Helios after replacing all instances of the Fiat-Shamir transformation with the weak Fiat-Shamir transformation and excluding the (optional) messages input to ProveCiph—i.e., ProveCiph should be used as a binary function. Helios 2.0 is  $\widehat{\text{Helios}}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ , where  $\Gamma$ ,  $\Sigma_1$ ,  $\Sigma_2$ ,  $\Sigma_3$  and  $\mathcal{H}$  are given in Definition 26.*

**Proposition 19.** *Helios 2.0 does not satisfy Ver-Ext.*

### Fig. 2 Adversary against Helios 2.0

Given a security parameter  $k$  as input,  $\mathcal{A}$  computes primes  $p$  and  $1$  such that  $p = 2 \cdot q + 1$  and  $q$  is of length  $k$ .  $\mathcal{A}$  also computes a generator  $g$  of the multiplicative group  $\mathbb{Z}_p^*$ . Let  $n_C \leftarrow 2$  and  $m \leftarrow \mathbb{N}_{q-1}$ , moreover, let  $m > 1$  be an element of  $m$ . The adversary proceeds as follows:

```

1 %coins
2  $(a_0, b_0, a_1, b_1) \leftarrow_R \mathbb{Z}_q^4$ ;
3 %witnesses
4  $A_0 \leftarrow g^{a_0} \pmod{p}$ ;
5  $B_0 \leftarrow g^{b_0} \pmod{p}$ ;
6  $A_1 \leftarrow g^{a_1} \pmod{p}$ ;
7  $B_1 \leftarrow g^{b_1} \pmod{p}$ ;
8 %challenge hash
9  $c \leftarrow \mathcal{H}(A_0, B_0, A_1, B_1) \pmod{q}$ ;
10 %private key
11  $x \leftarrow \frac{(b_0+c \cdot m) \cdot (1-m) - b_1 \cdot m}{a_0 \cdot (1-m) - a_1 \cdot m} \pmod{q}$ ;
12 %challenges
13  $c_1 \leftarrow \frac{b_1 - a_1 \cdot x}{1-m} \pmod{q}$ ;
14  $c_0 \leftarrow c - c_1 \pmod{q}$ ;
15 %coins
16  $r \leftarrow_R \mathbb{Z}_q$ ;
17 %responses
18  $f_0 \leftarrow a_0 + c_0 \cdot r \pmod{q}$ ;
19  $f_1 \leftarrow a_1 + c_1 \cdot r \pmod{q}$ ;
20 %proof of plaintext knowledge
21  $\sigma \leftarrow (A_0, B_0, c_0, f_0, A_1, B_1, c_1, f_1)$ ;
22 %public key
23  $h \leftarrow g^x \pmod{p}$ ;  $pk \leftarrow (p, q, g, h)$ ;
24 %proof of correct key construction
25  $\rho \leftarrow \text{ProveKey}((1^k, pk, m), (x, r'))$ ;
26 %ciphertext
27  $e \leftarrow (g^r \pmod{p}, h^r \cdot g^m \pmod{p})$ ;
28 %bulletin board
29  $BB \leftarrow \{(e, \sigma, \rho)\}$ ;
30 %tally
31  $\mathbf{X} \leftarrow (m, 1 - m)$ ;
32 %proof of decryption
33  $\mathbf{P} \leftarrow (\text{ProveDec}((pk, e, m), x))$ ;
34 return  $((pk, m, \rho), BB, n_C, \mathbf{X}, \mathbf{P})$ 

```

where  $r'$  is computed such that  $(pk, x, m) = \text{Gen}(1^k; r')$ .

Our proof of Proposition 19 formalizes the attack by Bernhard et al. [20, §3] in the context of our universal verifiability experiment.

*Proof.* Let Vote and Tally be the vote and tallying algorithms defined by Helios 2.0. Moreover, let  $wFS(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ ,  $wFS(\Sigma_2, \mathcal{H}) = (\text{ProveCiph}, \text{VerCiph})$  and  $wFS(\Sigma_3, \mathcal{H}) = (\text{ProveDec}, \text{VerDec})$ . We construct an adversary  $\mathcal{A}$  (Figure 2) against the universal verifiability experiment.

Suppose an execution of  $\text{Exp-UV-Ext}(\Pi, \mathcal{A}, k)$  computes

$$(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(k);$$

$$\mathbf{Y} \leftarrow \text{correct-tally}(pk, BB, n_C, k)$$

Since  $m > 1$ , there is no choice  $\beta \in \{1, 2\}$  nor coins  $r$  such that  $\text{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r) \in BB$ . By definition of function *correct-tally*, we have  $\mathbf{Y} = (0, 0)$ . Moreover, since  $\mathbf{X} = (m, 1 - m)$ , we have  $\mathbf{X} \neq \mathbf{Y}$  and  $\mathbf{X}[2] = 1 - \mathbf{X}[1]$ . Let us show that  $\text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$ . By definition of *Verify*, we have  $PK_{\mathcal{T}}$  is a vector  $(pk, m, \rho)$ . Moreover, by the completeness of  $(\text{ProveKey}, \text{VerKey})$  and  $(\text{ProveDec}, \text{VerDec})$ , we have  $\text{VerKey}((1^k, pk, m), \rho) = 1$  and  $\text{VerDec}((pk, e, \mathbf{X}[1]), P[1]) = 1$ . It remains to show that  $BB$  is the largest subset of  $BB$  satisfying the conditions given by the Tally algorithm. Since  $BB = \{(e, \sigma, \sigma)\}$  and  $(e, \sigma, \sigma)$  is a vector of length  $2 \cdot n_C - 1$ , it suffices to show that  $\text{VerCiph}((pk, e, \{0, 1\}), \sigma) = 1$ . Let us recall the definition of *VerCiph* (cf. [42, Figure 1] and Definition 27):

- $\text{VerCiph}((pk, e, \{0, 1\}), \sigma)$ . Parses  $pk$  as  $(p, q, g, h)$ ,  $e$  as  $(R, S)$ , and  $\sigma$  as  $(A_0, B_0, c_0, f_0, A_1, B_1, c_1, f_1)$ , outputting 0 if parsing fails. If  $g^{f_0} \equiv A_0 \cdot R^{c_0} \pmod{p} \wedge h^{f_0} \equiv B_0 \cdot S^{c_0} \pmod{p} \wedge g^{f_1} \equiv A_1 \cdot R^{c_1} \pmod{p} \wedge h^{f_1} \equiv B_1 \cdot (S/g)^{c_1} \pmod{p} \wedge \mathcal{H}(A_0, B_0, A_1, B_1) \equiv c_0 + c_1 \pmod{p}$ , then output 1, otherwise, output 0.

We have

$$\begin{aligned} g^{f_0} &\equiv g^{a_0 + c_0 \cdot r} \equiv g^{a_0} \cdot (g^r)^{c_0} \equiv A_0 \cdot R^{c_0} \pmod{p} \\ g^{f_1} &\equiv g^{a_1 + c_1 \cdot r} \equiv g^{a_1} \cdot (g^r)^{c_1} \equiv A_1 \cdot R^{c_1} \pmod{p} \end{aligned}$$

Moreover, we have  $h^{f_0} \equiv g^{x(a_0 + c_0 \cdot r)} \pmod{p}$  and  $B_0 \cdot S^{c_0} \equiv g^{b_0 + c_0(x \cdot r + m)} \pmod{p}$ , hence, to show  $h^{f_0} \equiv B_0 \cdot S^{c_0} \pmod{p}$ , it is sufficient to show  $(b_0 + c_0 \cdot m) \equiv x \cdot a_0 \pmod{q}$ :

$$\begin{aligned} &b_0 + c_0 \cdot m \\ &\equiv b_0 + c \cdot m - m \cdot c_1 \\ &\equiv b_0 + c \cdot m - \frac{b_1 \cdot m - a_1 \cdot m \cdot x}{1 - m} \\ &\equiv \frac{(b_0 + c \cdot m)(1 - m) - b_1 \cdot m + a_1 \cdot m \cdot x}{1 - m} \\ &\equiv \frac{(b_0 + c \cdot m)(1 - m) - b_1 \cdot m + \frac{a_1 \cdot m \cdot ((b_0 + c \cdot m)(1 - m) - b_1 \cdot m)}{a_0(1 - m) - a_1 \cdot m}}{1 - m} \\ &\equiv \frac{(a_0(1 - m) - a_1 \cdot m)((b_0 + c \cdot m)(1 - m) - b_1 \cdot m)}{(1 - m)(a_0(1 - m) - a_1 \cdot m)} \\ &\quad + \frac{a_1 \cdot m \cdot ((b_0 + c \cdot m)(1 - m) - b_1 \cdot m)}{(1 - m)(a_0(1 - m) - a_1 \cdot m)} \\ &\equiv \frac{a_0(1 - m)((b_0 + c \cdot m)(1 - m) - b_1 \cdot m)}{(1 - m)(a_0(1 - m) - a_1 \cdot m)} \\ &\equiv \frac{a_0 \cdot ((b_0 + c \cdot m)(1 - m) - b_1 \cdot m)}{a_0(1 - m) - a_1 \cdot m} \\ &\equiv x \cdot a_0 \pmod{q} \end{aligned}$$

Similarly,  $h^{f_1} \equiv g^{x(a_1 + c_1 \cdot r)} \pmod{p}$  and  $B_1 \cdot (S/g)^{c_1} \equiv g^{b_1 + c_1(x \cdot r + m - 1)} \pmod{p}$ , hence, to show  $h^{f_1} \equiv B_1 \cdot (S/g)^{c_1} \pmod{p}$ , it is sufficient to show  $b_1 + c_1(m - 1) \equiv a_1 \cdot x \pmod{q}$ :

$$\begin{aligned} &b_1 + c_1(m - 1) \\ &\equiv b_1 + \frac{(m - 1)(b_1 - a_1 \cdot x)}{1 - m} \\ &\equiv \frac{b_1(1 - m) + (m - 1)(b_1 - a_1 \cdot x)}{1 - m} \\ &\equiv \frac{a_1 \cdot x(1 - m)}{1 - m} \\ &\equiv a_1 \cdot x \pmod{q} \end{aligned}$$

Furthermore, we have

$$\begin{aligned} \mathcal{H}(A_0, B_0, A_1, B_1) &\equiv c_0 + c_1 \equiv c - c_1 + c_1 \\ &\equiv \mathcal{H}(A_0, B_0, A_1, B_1) - c_1 + c_1 \pmod{p} \end{aligned}$$

It follows that  $\text{VerCiph}((pk, e, \{0, 1\}), \sigma) = 1$ , concluding our proof.  $\square$

## APPENDIX F

### PROOF: Exp-EV-Int $\Rightarrow$ Exp-IV-Int

Our eligibility verifiability experiment (§IV-B3) asserts that no one can construct a ballot that appears to be associated with public credential  $pk$  unless they know private credential  $sk$ . It follows that a voter can uniquely identify her ballot on the bulletin board, because no one else knows her private credential. Eligibility verifiability therefore implies individual verifiability (Theorem 4).

Our proof of Theorem 4 is reliant on distinct credentials, which is an consequence of eligibility verifiability:

**Lemma 20.** *If an election scheme  $\Pi$  satisfies strong eligibility verifiability, then there exists a negligible function  $\mu$ , such that for all security parameters  $k$ , we have*

$$\begin{aligned} &Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k); \\ &\quad (pk_0, sk_0) \leftarrow \text{Register}(PK_{\mathcal{T}}, k); \\ &\quad (pk_1, sk_1) \leftarrow \text{Register}(PK_{\mathcal{T}}, k) : \\ &\quad sk_0 = sk_1] \leq \mu(k) \end{aligned}$$

*Proof.* Suppose an election scheme  $\Pi$  satisfies Exp-EV-Int, but

$$\begin{aligned} &Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k); \\ &\quad (pk_0, sk_0) \leftarrow \text{Register}(PK_{\mathcal{T}}, k); \\ &\quad (pk_1, sk_1) \leftarrow \text{Register}(PK_{\mathcal{T}}, k) : \\ &\quad sk_0 = sk_1] \geq \frac{1}{p(k)} \end{aligned}$$

for some polynomial  $p$  and security parameter  $k$ . Then we can construct an adversary  $\mathcal{A}$  that wins Exp-EV-Int as follows. Adversary  $\mathcal{A}$  is given input  $k$  and runs *Setup* to obtain a key pair  $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$ , chooses some positive integer  $n_V$ , and outputs  $(PK_{\mathcal{T}}, n_V)$ . The challenger then generates  $n_V$  key pairs and gives the set  $L$  of public keys to  $\mathcal{A}$ . Now  $\mathcal{A}$  simply runs *Register* $(PK_{\mathcal{T}}, k)$  to get a key pair  $(pk, sk)$ , chooses some positive integers  $n_C$  and  $\beta$  such that  $1 \leq \beta \leq n_C$ , computes  $b \leftarrow \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$ , and outputs  $(n_C, b)$ . We know that secret keys generated by *Register* collide with probability at least  $\frac{1}{p(k)}$ , so *Register* must generate a particular secret key  $sk'$  with probability  $\frac{1}{p(k)}$ . Therefore, this  $sk'$  will correspond to one of the public keys in  $L$  with probability  $\frac{n_V}{p(k)}$ . Furthermore, the key  $sk$  generated by the adversary will be  $sk'$  with probability  $\frac{1}{p(k)}$ . Therefore,  $b$  will be a vote constructed under a voter's secret key with probability  $\frac{n_V}{p(k)^2}$ , so  $\mathcal{A}$  wins the experiment with non-negligible probability.  $\square$

### A. Proof: Theorem 4

Suppose there exists an adversary  $\mathcal{A}'$  that wins  $\text{Exp-IV-Int}(\Pi, \mathcal{A}', k)$  with probability  $\frac{1}{p(k)}$  for some polynomial  $p$ . Then we can construct an adversary  $\mathcal{A}$  that wins  $\text{Exp-EV-Int}(\Pi, \mathcal{A}, k)$  with non-negligible probability. Adversary  $\mathcal{A}$  is given  $k$  as input, which it passes to  $\mathcal{A}'$ . Adversary  $\mathcal{A}'$  may ask for secret keys from its oracle  $C$ , in which case  $\mathcal{A}$  forwards these queries to its own, identical oracle. Adversary  $\mathcal{A}$  then forwards the oracle's response back to  $\mathcal{A}'$ . Adversary  $\mathcal{A}'$  then outputs  $(PK_{\mathcal{T}}, n_V)$ , which is then output by  $\mathcal{A}$ . Next,  $\mathcal{A}$  is given the public keys  $(pk_1, \dots, pk_{n_V})$ . Adversary  $\mathcal{A}$  passes these keys to  $\mathcal{A}'$ , which returns  $(n_C, \beta, \beta', i, j)$ . Any oracle queries made by  $\mathcal{A}'$  are handled exactly as before. Now  $\mathcal{A}$  queries its oracle  $C$  on  $i$ . The oracle returns  $sk_i$ . Adversary  $\mathcal{A}$  computes  $b = \text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta)$  and outputs  $(n_C, \beta', j, b)$ . Adversary  $\mathcal{A}'$  wins  $\text{Exp-IV-Int}(\Pi, \mathcal{A}, k)$  with non-negligible probability, so with non-negligible probability  $b = \text{Vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta')$  and  $\mathcal{A}'$  (and therefore  $\mathcal{A}$ ) did not query the oracle on input  $j$ . Adversary  $\mathcal{A}$  only makes one additional oracle query on input  $i$ , so again,  $\mathcal{A}$  does not query the oracle on  $j$ . Furthermore, by Lemma 20,  $sk_i = sk_j$  with only negligible probability. Therefore  $\mathcal{A}$  wins  $\text{Exp-EV-Int}(\Pi, \mathcal{A}, k)$  with probability  $\frac{1}{p(k)} - \text{negl}(k)$ .  $\square$

### APPENDIX G JCJ SCHEME

We formalize a generic construction for JCJ-like election schemes (Figure 3). Our construction is parameterized on the choice of homomorphic encryption scheme and sigma protocols.<sup>32</sup> The specification of algorithms Setup, Register and Vote follow from our informal descriptions (§V).<sup>33</sup> The tallying algorithm performs the following steps:

- 1) *Remove invalid ballots*: The tallier discards any ballots from the bulletin board for which proofs do not hold.
- 2) *Eliminating duplicates*: The tallier performs pairwise PETs on the encrypted credentials and discard any ballots for which a test holds, that is, ballots using the same credential are discarded.<sup>34</sup>
- 3) *Mixing*: The tallier mixes the ciphertexts in the ballots (i.e., the encrypted choices and the encrypted credentials), using the same secret permutation for both mixes, hence, the mix preserves the relation between encrypted choices and credentials. Let  $\mathbf{C}_1$  and  $\mathbf{C}_2$  be the vectors output by these mixes. The tallier also mixes the public credentials published by the registrar. Let  $\mathbf{C}_3$  be the vector output by this mix.
- 4) *Remove ineligible ballots*: The tallier discards ciphertexts  $\mathbf{C}_1[i]$  from  $\mathbf{C}_1$  if there is no ciphertext  $c$  in  $\mathbf{C}_3$  such that

<sup>32</sup>For brevity, the encryption scheme's message space  $m$  is assumed to contain  $\{1, \dots, |m|\}$ .

<sup>33</sup>Algorithm Setup bounds the maximum number of voters to a polynomial in the security parameter to ensure that private voter credentials do not collide, with overwhelming probability.

<sup>34</sup>JCJ permits revoting; ballots are removed in accordance with a revoting policy [79, §4.1]. Since election schemes that permit revoting cannot satisfy our definition of universal verifiability (§IV-B2), we assume that the revoting policy forbids revoting, i.e., ballots using the same credential are discarded.

a PET holds for  $c$  and  $\mathbf{C}_2[i]$ , that is, ballots cast using ineligible credentials are discarded.

- 5) *Decrypting*: The tallier decrypts the remaining encrypted choices in  $\mathbf{C}_1$  and proves that decryption was performed correctly. The tallier identifies the winning candidate from the decrypted choices.

The Verify algorithm checks that each of the above steps has been performed correctly.

Lemmata 21–23 demonstrate that generalized JCJ is a construction for election schemes.

**Lemma 21.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6$  and  $\mathcal{H}$  satisfy the preconditions of Figure 3. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies Correctness.*

*Proof.* Our proof is by induction on the number of ballots  $n_B$ . We start with the base case,  $n_B = 1$ . For all  $k, n_C$ , and  $\beta \in \{1, \dots, n_C\}$ , we have

$$\begin{aligned} (PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) &\leftarrow \text{Setup}(k); \\ (pk, sk) &\leftarrow \text{Register}(PK_{\mathcal{T}}, k); \\ b &\leftarrow \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k); \\ \mathbf{Y}[\beta] &\leftarrow \mathbf{Y}[\beta] + \mathbf{1}; \\ L &\leftarrow \{pk\}; \\ BB &\leftarrow \{b\}; \\ (\mathbf{X}, P) &\leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k); \end{aligned}$$

Assume  $n_C \leq m_C$  (otherwise, we trivially satisfy correctness). Hence, we need to show  $\mathbf{X}[\beta] = 1$  and  $\mathbf{X}[i] = 0$  for all  $i \neq \beta$ . By definition of Setup, we have  $PK_{\mathcal{T}} = (pk_T, m, \rho)$  and  $m_C = |m|$ . By definition of Vote, we have  $b = (c_1, c_2, \sigma, \tau)$ , where  $c_1 = \text{Enc}(pk_T, \beta; r_1)$ ,  $c_2 = \text{Enc}(pk_T, sk; r_2)$ ,  $\sigma = \text{ProveCiph}((pk_T, c_1, \{1, \dots, n_C\}), (\beta, r_1))$ , and  $\tau = \text{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, sk, r_2))$ . Since  $\beta \in \{1, \dots, n_C\}$  and  $n_C \leq |m|$ , we have  $\beta$  is a message in  $\Gamma$ 's message space

- *Remove invalid ballots*: This involves checking the proofs  $\sigma$  and  $\tau$ . Since they were honestly computed, they verify with overwhelming probability.
- *Remove duplicate ballots*: Tally would check here if there are multiple ballots computed using the same secret key. Since there is only one ballot, this check passes trivially.
- *Mixing*: Tally mixes the ballots. Since there is only one ballot, Tally will just re-encrypt the ballot. Let the re-encryptions of  $b[1]$  and  $b[2]$  be  $b'[1]$  and  $b'[2]$ , respectively. This is done honestly, so  $b'[1]$  will still be an encryption of  $\beta$  and  $b'[2]$  will still be an encryption of  $sk$ .
- *Remove ineligible ballots*: As mentioned,  $b'[2]$  is still an encryption of  $sk$ , which is a valid secret key, so the ballot is not eliminated.
- *Decrypting*: Finally, Tally computes  $\beta' \leftarrow \text{Dec}(pk_T, SK_{\mathcal{T}}, b'[1])$ . Again, since  $b'[1]$  is still an encryption of  $\beta$ , we have  $\beta' = \beta$ . Tally then increments  $\mathbf{X}[\beta]$  by 1.

Since we now have  $\mathbf{X}[\beta] = 1$  and  $\mathbf{X}[i] = 0$  for all  $i \neq \beta$ , we have that JCJ satisfies correctness when  $n_B = 1$ .

**Fig. 3** Generalized JCJ

Suppose  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$  is a multiplicatively homomorphic asymmetric encryption scheme with a message space over  $\mathbb{Z}_m^*$  for some integer  $m$  determined by the security parameter,  $\epsilon$  is an identity element of  $\Gamma$ 's message space with respect to  $\odot$ ,  $\Sigma_1$  proves correct key construction,  $\Sigma_2$  proves plaintext knowledge in a subspace,  $\Sigma_3$  proves conjunctive plaintext knowledge,  $\Sigma_4$  proves correct decryption,  $\Sigma_5$  is a PET,  $\Sigma_6$  is a mixnet, and  $\mathcal{H}$  is a hash function. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ ,  $\text{FS}(\Sigma_2, \mathcal{H}) = (\text{ProveCiph}, \text{VerCiph})$ ,  $\text{FS}(\Sigma_3, \mathcal{H}) = (\text{ProveBind}, \text{VerBind})$ ,  $\text{FS}(\Sigma_4, \mathcal{H}) = (\text{ProveDec}, \text{VerDec})$ ,  $\text{FS}(\Sigma_5, \mathcal{H}) = (\text{ProvePET}, \text{VerPET})$ , and  $\text{FS}(\Sigma_6, \mathcal{H}) = (\text{ProveMix}, \text{VerMix})$ . We define *generalized JCJ*  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H}) = (\text{Setup}, \text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  as follows.

- **Setup**( $k$ ). Select coins  $r$ , compute  $(pk_T, sk_T, m) \leftarrow \text{Gen}(1^k; r); \rho \leftarrow \text{ProveKey}((1^k, pk_T, m), (sk_T, r)); PK_T \leftarrow (pk_T, m, \rho); m_C \leftarrow |m|$ , and output  $(PK_T, sk_T, \text{poly}(k), m_C)$ .
- **Register**( $PK_T, k$ ). Parse  $PK_T$  as  $(pk_T, m, \rho)$ , outputting  $(\perp, \perp)$  if parsing fails or  $\text{VerKey}((1^k, pk_T, m), \rho) \neq 1$ . Compute  $d \leftarrow_R m; pd \leftarrow \text{Enc}(pk_T, d)$  and output  $(d, pd)$ .
- **Vote**( $d, PK_T, n_C, \beta, k$ ). Parse  $PK_T$  as a vector  $(pk_T, m, \rho)$ , outputting  $\perp$  if parsing fails or  $\text{VerKey}((1^k, pk_T, m), \rho) \neq 1 \vee \beta \notin \{1, \dots, n_C\} \vee \{1, \dots, n_C\} \not\subseteq m$ . Select coins  $r_1$  and  $r_2$ , compute  $c_1 \leftarrow \text{Enc}(pk_T, \beta; r_1); c_2 \leftarrow \text{Enc}(pk_T, d; r_2); \sigma \leftarrow \text{ProveCiph}((pk_T, c_1, \{1, \dots, n_C\}), (\beta, r_1)); \tau \leftarrow \text{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, d, r_2))$  and output ballot  $(c_1, c_2, \sigma, \tau)$ .
- **Tally**( $PK_T, sk_T, BB, L, n_C, k$ ). Initialize vectors  $\mathbf{X}$  of length  $n_C$  and  $\mathbf{P}$  of length 9. Parse  $PK_T$  as  $(pk_T, m, \rho)$ . Compute **for**  $1 \leq j \leq n_C$  **do**  $\mathbf{X}[j] \leftarrow 0$ . Proceed as follows.
  - 1) *Remove invalid ballots*: Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a vector of length 4 and  $\text{VerCiph}((pk_T, b_i[1], \{1, \dots, n_C\}), b_i[3]) = 1 \wedge \text{VerBind}((pk_T, b_i[1], b_i[2], b_i[4]) = 1$ . If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then output  $(\mathbf{X}, \mathbf{P})$ .
  - 2) *Eliminating duplicates*: Initialize  $\mathbf{P}_{\text{dupl}}$  as a vector of length  $\ell$ . For each  $1 \leq i \leq \ell$ , if there exists  $\sigma$  and  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  such that  $\sigma \leftarrow \text{ProvePET}((pk_T, b_i[2], b_j[2], 1), sk_T)$  and  $\text{VerPET}((pk_T, b_i[2], b_j[2], 1), \sigma) = 1$ , then assign  $\mathbf{P}_{\text{dupl}}[i] \leftarrow (\sigma)$ ; otherwise, compute  $\sigma_j \leftarrow \text{ProvePET}((pk_T, b_i[2], b_j[2], 0), sk_T)$  for each  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  and assign  $\mathbf{P}_{\text{dupl}}[i] \leftarrow (\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_\ell)$ . Let  $\mathbf{BB}$  be the empty vector and compute **for**  $1 \leq i \leq \ell \wedge |\mathbf{P}_{\text{dupl}}[i]| = \ell - 1$  **do**  $\mathbf{BB} \leftarrow \mathbf{BB} \parallel (b_i)$ , where  $\mathbf{BB} \parallel (b_i)$  denotes the concatenation of vectors  $\mathbf{BB}$  and  $(b_i)$ , i.e.,  $\mathbf{BB} \parallel (b_i) = (\mathbf{BB}[1], \dots, \mathbf{BB}[|\mathbf{BB}|], b_i)$ .
  - 3) *Mixing*: Suppose  $\mathbf{BB} = (b'_1, \dots, b'_{|\mathbf{BB}|})$ . Select a random permutation  $\chi$  on  $\{1, \dots, |\mathbf{BB}|\}$ , initialize  $\mathbf{C}_1, \mathbf{C}_2, \mathbf{r}_1$  and  $\mathbf{r}_2$  as vectors of length  $|\mathbf{BB}|$ , and fill  $\mathbf{r}_1$  and  $\mathbf{r}_2$  with random coins. Compute **for**  $1 \leq i \leq |\mathbf{BB}|$  **do**  $\mathbf{C}_1[\chi(i)] \leftarrow b'_i[1] \otimes \text{Enc}(PK_T, \epsilon; \mathbf{r}_1[i]); \mathbf{C}_2[\chi(i)] \leftarrow b'_i[2] \otimes \text{Enc}(PK_T, \epsilon; \mathbf{r}_2[i])$  and  $P_{\text{mix},1} \leftarrow \text{ProveMix}((pk_T, (b'_1[1], \dots, b'_{|\mathbf{BB}|}[1]), \mathbf{C}_1), (\mathbf{r}_1, \chi)); P_{\text{mix},2} \leftarrow \text{ProveMix}((pk_T, (b'_1[2], \dots, b'_{|\mathbf{BB}|}[2]), \mathbf{C}_2), (\mathbf{r}_2, \chi))$ . Similarly, select a random permutation  $\chi'$  on  $\{1, \dots, |L|\}$ , initialize  $\mathbf{C}_3$  and  $\mathbf{r}_3$  as vectors of length  $|L|$ , fill  $\mathbf{r}_3$  with random coins, and compute **for**  $1 \leq i \leq |L|$  **do**  $\mathbf{C}_3[\chi'(i)] \leftarrow L[i] \otimes \text{Enc}(PK_T, \epsilon; \mathbf{r}_3[i])$  and  $P_{\text{mix},3} \leftarrow \text{ProveMix}((pk_T, L), (\mathbf{r}_3, \chi'))$ .
  - 4) *Remove ineligible ballots*: Initialize  $\mathbf{P}_{\text{inelig}}$  as a vector of length  $|\mathbf{C}_2|$ . For each  $1 \leq i \leq |\mathbf{C}_2|$ , if there exists  $\sigma$  and  $c \in \mathbf{C}_3$  such that  $\sigma \leftarrow \text{ProvePET}((pk_T, \mathbf{C}_2[i], c, 1), sk_T)$  and  $\text{VerPET}((pk_T, \mathbf{C}_2[i], c), \sigma) = 1$ , then assign  $\mathbf{P}_{\text{inelig}}[i] \leftarrow (\sigma)$ ; otherwise, compute  $\sigma_j \leftarrow \text{ProvePET}((pk_T, \mathbf{C}_2[i], \mathbf{C}_3[j], 0), sk_T)$  for each  $j \in \{1, \dots, |\mathbf{C}_3|\}$  and assign  $\mathbf{P}_{\text{inelig}}[i] \leftarrow (\sigma_1, \dots, \sigma_{|\mathbf{C}_3|})$ .
  - 5) *Decrypting*: Initialize  $\mathbf{P}_{\text{dec}}$  as the empty vector. Compute **for**  $1 \leq i \leq |\mathbf{C}_1| \wedge |\mathbf{P}_{\text{inelig}}[i]| = 1$  **do**  $\beta \leftarrow \text{Dec}(pk_T, sk_T, \mathbf{C}_1[i]); \sigma \leftarrow \text{ProveDec}((pk_T, \mathbf{C}_1[i], \beta), sk_T); \mathbf{X}[\beta] \leftarrow \mathbf{X}[\beta] + 1; \mathbf{P}_{\text{dec}} \leftarrow \mathbf{P}_{\text{dec}} \parallel (\sigma)$ . Assign  $\mathbf{P} \leftarrow (\mathbf{P}_{\text{dupl}}, \mathbf{C}_1, P_{\text{mix},1}, \mathbf{C}_2, P_{\text{mix},2}, \mathbf{C}_3, P_{\text{mix},3}, \mathbf{P}_{\text{inelig}}, \mathbf{P}_{\text{dec}})$  and output  $(\mathbf{X}, \mathbf{P})$ .
- **Verify**( $PK_T, BB, L, n_C, \mathbf{X}, \mathbf{P}, k$ ). Parse  $PK_T$  as a vector  $(pk_T, m, \rho)$ ,  $\mathbf{X}$  as a vector of length  $n_C$ , and  $\mathbf{P}$  as a vector  $(\mathbf{P}_{\text{dupl}}, \mathbf{C}_1, P_{\text{mix},1}, \mathbf{C}_2, P_{\text{mix},2}, \mathbf{C}_3, P_{\text{mix},3}, \mathbf{P}_{\text{inelig}}, \mathbf{P}_{\text{dec}})$ , outputting 0 if parsing fails or  $\text{VerKey}((1^k, pk_T, m), \rho) \neq 1$ . Let  $m_C = |m|$ . If  $n_C > m_C$ , then output 0. Otherwise, perform the following checks:
  - 1) *Check removal of invalid ballots*: Compute  $\{b_1, \dots, b_\ell\}$  as per Step (1) of the tallying algorithm. If  $\{b_1, \dots, b_\ell\} = \emptyset$  and  $\mathbf{X}$  is a zero-filled vector, then output 1. Otherwise, proceed as follows.
  - 2) *Check duplicate elimination*: Check that  $\mathbf{P}_{\text{dupl}}$  is a vector of length  $\ell$  and that for all  $1 \leq i \leq \ell$ , either: i)  $|\mathbf{P}_{\text{dupl}}[i]| = 1$  and there exists  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  such that  $\text{VerPET}((pk_T, b_i[2], b_j[2], 1), \mathbf{P}_{\text{dupl}}[i][1]) = 1$ , or ii)  $|\mathbf{P}_{\text{dupl}}[i]| = \ell - 1$  and for all  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  we have  $\text{VerPET}((pk_T, b_i[2], b_j[2], 0), \mathbf{P}_{\text{dupl}}[i][j]) = 1$ .
  - 3) *Check mixing*: Compute  $\mathbf{BB}$  as per Step (2) of the tallying algorithm, suppose  $\mathbf{BB} = (b'_1, \dots, b'_{|\mathbf{BB}|})$ , and check that  $\text{VerMix}((pk_T, (b'_1[1], \dots, b'_{|\mathbf{BB}|}[1]), \mathbf{C}_1), P_{\text{mix},1}) = 1 \wedge \text{VerMix}((pk_T, (b'_1[2], \dots, b'_{|\mathbf{BB}|}[2]), \mathbf{C}_2), P_{\text{mix},2}) = 1 \wedge \text{VerMix}((pk_T, L, \mathbf{C}_3), P_{\text{mix},3}) = 1$ .
  - 4) *Check removal of ineligible ballots*: Check that  $\mathbf{P}_{\text{inelig}}$  is a vector of length  $|\mathbf{C}_2|$  and that for all  $1 \leq i \leq |\mathbf{C}_2|$ , either: i)  $|\mathbf{P}_{\text{inelig}}[i]| = 1$  and there exists  $c \in \mathbf{C}_3$  such that  $\text{VerPET}((pk_T, \mathbf{C}_2[i], c, 1), \mathbf{P}_{\text{inelig}}[i][1]) = 1$ , or ii)  $|\mathbf{P}_{\text{inelig}}[i]| = |\mathbf{C}_3|$  and for all  $1 \leq j \leq |\mathbf{C}_3|$  we have  $\text{VerPET}((pk_T, \mathbf{C}_2[i], \mathbf{C}_3[j], 0), \mathbf{P}_{\text{inelig}}[i][j]) = 1$ .
  - 5) *Check decryption*: Compute  $\mathbf{C}'_1$  as follows:  $\mathbf{C}'_1 \leftarrow ()$ ; **for**  $1 \leq i \leq |\mathbf{C}_1| \wedge |\mathbf{P}_{\text{inelig}}[i]| = 1$  **do**  $\mathbf{C}'_1 \leftarrow \mathbf{C}'_1 \parallel (\mathbf{C}_1[i])$ . Check that there exists  $\beta_1, \dots, \beta_{|\mathbf{C}'_1|}$  such that  $\mathbf{X}[i] = \{j : 1 \leq j \leq |\mathbf{C}'_1| \wedge \beta_j = i\}$  and for all  $1 \leq i \leq |\mathbf{C}'_1|$  we have  $\text{VerDec}((pk_T, \mathbf{C}'_1[i], \beta_i), \mathbf{P}_{\text{dec}}[i]) = 1$ .

Now we assume that JCJ is correct for  $n_B = n$ , and prove that it satisfies correctness for  $n_B = n + 1$ . First, we note that since we are only adding one more vote, and therefore only registering one more key pair, the probability that  $sk_{n+1} = sk_i$  for some  $i \in \{1, \dots, n_B\}$  is negligible, since JCJ ensures that  $n_B$  is bounded by a polynomial in  $k$  and the secret keys are just random nonces. Now it is easy to see that the only step of Tally that we need to be concerned about is the step in which duplicate ballots are removed. This is because the checks performed in the other steps all pass with overwhelming probability when the computation is done honestly. In the step to remove duplicate ballots, we need to make sure that there are not multiple ballots computed using  $sk_{n+1}$ . As we argued above,  $sk_{n+1}$  is unique among the secret keys, so the ballot computed using  $sk_{n+1}$  will not be removed, and we will get that  $\mathbf{X} = \mathbf{Y}$ . Therefore, JCJ satisfies correctness.  $\square$

**Lemma 22.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6$  and  $\mathcal{H}$  satisfy the preconditions of Figure 3. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies Completeness.*

*Proof.* Let  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H}) = (\text{Setup}, \text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$ ,  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ ,  $\text{FS}(\Sigma_4, \mathcal{H}) = (\text{ProveDec}, \text{VerDec})$ ,  $\text{FS}(\Sigma_5, \mathcal{H}) = (\text{ProvePET}, \text{VerPET})$ , and  $\text{FS}(\Sigma_6, \mathcal{H}) = (\text{ProveMix}, \text{VerMix})$ . Suppose  $k$  is a security parameter,  $BB$  is a bulletin board, and  $n_C$  is an integer. Further suppose  $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$  is a key pair,  $m_B$  and  $m_C$  are integers,  $L$  is an electoral roll (i.e., a set of public keys output by Register), and  $(\mathbf{X}, P)$  is a tally, such that  $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k)$  and  $(\mathbf{X}, P) \leftarrow \text{Tally}(PK_{\mathcal{T}}, SK_{\mathcal{T}}, BB, L, n_C, k)$ . Moreover, suppose  $n_C \leq m_C$ . By definition of Setup, there exist coins  $r$  such that  $(pk, SK_{\mathcal{T}}, m) = \text{Gen}(1^k; r)$ ,  $PK_{\mathcal{T}} \leftarrow (pk, m, \rho)$ , and  $m_C = |m|$ , where  $\rho$  is an output of  $\text{ProveKey}(1^k, PK_{\mathcal{T}}, m)$ ,  $(SK_{\mathcal{T}}, r)$ . Since  $n_C$  is at most  $|m|$ , we have that any  $\beta \in \{1, \dots, n_C\}$  is in  $\Gamma$ 's message space. Moreover, by the definition of Tally, vector  $\mathbf{X}$  is of length  $n_C$  and  $P$  is a vector  $(\mathbf{P}_{\text{dupl}}, \mathbf{C}_1, P_{\text{mix},1}, \mathbf{C}_2, P_{\text{mix},2}, \mathbf{C}_3, P_{\text{mix},3}, \mathbf{P}_{\text{inelig}}, \mathbf{P}_{\text{dec}})$ . It follows that Verify can parse  $P$  and  $\mathbf{X}$  successfully. Moreover, by completeness of  $(\text{ProveKey}, \text{VerKey})$ , we have  $\text{VerKey}(1^k, pk, m, \rho) = 1$  with overwhelming probability. Suppose  $\{b_1, \dots, b_\ell\}$  is the largest subset of  $BB$  satisfying the conditions given by algorithm Tally. If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then  $\mathbf{X}$  is a zero-filled vector and Verify accepts, concluding our proof. Otherwise, we proceed by showing that checks (2)–(5) of Verify succeed:

- *Check duplicate elimination.* The check succeeds by completeness of  $(\text{ProvePET}, \text{VerPET})$ , namely, for all  $1 \leq i \leq \ell$  we have either: i)  $|\mathbf{P}_{\text{dupl}}[i]| = 1$  and there exists  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  such that  $\text{VerPET}((PK_{\mathcal{T}}, b_i[2], b_j[2], 1), \mathbf{P}_{\text{dupl}}[i][1]) = 1$ ; or ii)  $|\mathbf{P}_{\text{dupl}}[i]| = \ell - 1$  and for all  $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  we have  $\text{VerPET}((PK_{\mathcal{T}}, b_i[2], b_j[2], 0), \mathbf{P}_{\text{dupl}}[i][j]) = 1$ .
- *Check mixing.* Suppose  $BB = (b'_1, \dots, b'_{|BB|})$ . Then by the completeness of  $(\text{ProveMix}, \text{VerMix})$ , we have that

$$\begin{aligned} \text{VerMix}((PK_{\mathcal{T}}, (b'_1[1], \dots, b'_{|BB|}[1]), \mathbf{C}_1), P_{\text{mix},1}) &= 1 \wedge \\ \text{VerMix}((PK_{\mathcal{T}}, (b'_1[2], \dots, b'_{|BB|}[2]), \mathbf{C}_2), P_{\text{mix},2}) &= 1 \wedge \\ \text{VerMix}((PK_{\mathcal{T}}, L, \mathbf{C}_3), P_{\text{mix},3}) &= 1. \end{aligned}$$

- *Check removal of ineligible ballots.* By Step (4) of Tally, we have  $\mathbf{P}_{\text{inelig}}$  is a vector of length  $|\mathbf{C}_2|$ . Moreover, by completeness of  $(\text{ProvePET}, \text{VerPET})$ , for all  $1 \leq i \leq |\mathbf{C}_2|$  we have either: i)  $|\mathbf{P}_{\text{inelig}}[i]| = 1$  and there exists  $c \in \mathbf{C}_3$  such that  $\text{VerPET}((PK_{\mathcal{T}}, \mathbf{C}_2[i], c, 1), \mathbf{P}_{\text{inelig}}[i][1]) = 1$ ; or ii)  $|\mathbf{P}_{\text{inelig}}[i]| = |\mathbf{C}_3|$  and for all  $1 \leq j \leq |\mathbf{C}_3|$  we have  $\text{VerPET}((PK_{\mathcal{T}}, \mathbf{C}_2[i], \mathbf{C}_3[j], 0), \mathbf{P}_{\text{inelig}}[i][j]) = 1$ . It follows that the check succeeds.
- *Check decryption.* Verify computes the set  $\mathbf{C}'_1$  such that it includes only elements  $c_i$  of  $\mathbf{C}_1$  for which  $|\mathbf{P}_{\text{inelig}}[i]| = 1$ . Then, by the definition of Tally and the completeness of  $(\text{ProveDec}, \text{VerDec})$ , we have that  $\text{VerDec}((PK_{\mathcal{T}}, \mathbf{C}'_1[i], \beta_i), \mathbf{P}[9][i]) = 1$  for all  $1 \leq i \leq |\mathbf{C}'_1|$ . Furthermore, in step 5 of Tally, ballots  $\mathbf{C}_1[i]$  are only counted for a candidate when  $1 \leq i \leq |\mathbf{C}_1| \wedge |\mathbf{P}_{\text{inelig}}[i]| = 1$ , which is exactly how  $\mathbf{C}'_1$  is defined. Therefore, there exists  $\beta_1, \dots, \beta_{|\mathbf{C}'_1|}$  such that  $\mathbf{X}[i] = |\{j : 1 \leq j \leq |\mathbf{C}'_1| \wedge \beta_j = i\}|$ .

It follows that all the required checks succeed and Verify outputs 1, concluding our proof.  $\square$

**Lemma 23.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6$  and  $\mathcal{H}$  satisfy the preconditions of Figure 3. Further suppose  $\Gamma$  is collision-free. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies Injectivity.*

The proof of Lemma 23 is similar to the proof of Lemma 14.

*Proof sketch.* Generalized JCJ ballots contain encrypted choices, hence, collision-freeness of the encryption scheme ensures that distinct choices are not mapped to the same ballot.  $\square$

Generalized JCJ can be instantiate to derive JCJ:

**Definition 29 (JCJ).** JCJ [79] is  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ , where  $\Gamma$  is a modified version of El Gamal [51] invented by Juels et al. [79, §4] that can be seen as a simplified version of Cramer–Shoup [44],  $\Sigma_1$  is the proof of key construction by Gennaro et al. [57],  $\Sigma_4$  is the conjunction [41] of two Schnorr proofs [104],  $\Sigma_5$  is the PET by MacKenzie et al. [94],  $\Sigma_6$  is either the mixnet of Furukawa and Sako [54] or Neff [96], and  $\mathcal{H}$  is a random oracle. Juels et al. leave  $\Sigma_2$  and  $\Sigma_3$  unspecified.

Juels et al. [79] do not mandate particular cryptographic primitives, so Definition 29 might be seen more as an instantiation of their scheme than an exact recollection of it. We assume that the primitives in Definition 29 satisfy the properties required by generalized JCJ. We also assume that the sigma protocols satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 9 is applicable.

To show that JCJ is an election scheme, we must demonstrate that Correctness, Completeness and Injectivity are satisfied. Correctness follows immediately from Lemma 21 and

Completeness follows from Lemma 22. We show that Injectivity is also satisfied.

A non-interactive proof system derived from a sigma protocol for proving correct key construction is sufficient to ensure that El Gamal is collision-free:

**Lemma 24.** *Suppose  $\Sigma_1$  is a sigma protocol that proves correct key construction and  $\mathcal{H}$  is a hash function. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . Further suppose for all security parameters  $k$ , public keys  $pk$ , message spaces  $\mathfrak{m}$  and proofs  $\rho$ , we have  $\text{VerKey}((1^k, pk, \mathfrak{m}), \rho) = 1$  implies  $h \neq 0$ , there exists  $p, q, g$  and  $h$  such that  $pk = (p, q, g, h)$  and  $(p, q, g)$  are cryptographic parameters, and  $\mathfrak{m} = \{1, \dots, p-1\}$ . It follows that multiplicatively homomorphic El Gamal is collision-free for  $m_1, m_2 \in \mathfrak{m}$ .*

The proof of Lemma 24 is similar to the proof of Lemma 15.

*Proof.* Suppose  $k$  is a security parameter,  $pk$  is a public key,  $\mathfrak{m}$  is a message space,  $\rho$  is a proof,  $m_1, m_2 \in \mathfrak{m}$  are messages and  $r_1$  and  $r_2$  are coins such that  $\text{VerKey}((1^k, pk, \mathfrak{m}), \rho) = 1$ ,  $m_1 \neq m_2 \vee r_1 \neq r_2$ ,  $\mathfrak{m} = \{1, \dots, p-1\}$ , and  $pk = (p, q, g, h)$  for some  $p, q, g$  and  $h$ . Further suppose that  $c_1$  and  $c_2$  are ciphertexts such that  $c_1 = \text{Enc}(pk, m_1; r_1)$ ,  $c_2 = \text{Enc}(pk, m_2; r_2)$ , and  $\text{Enc}$  is El Gamal's encryption algorithm. If  $r_1 \neq r_2$ , then we proceed as follows. By definition of  $\text{Enc}$ , we have  $c_1[1] = g^{r_1} \pmod{p}$  and  $c_2[1] = g^{r_2} \pmod{p}$ . Since  $r_1$  and  $r_2$  are distinct, we have  $g^{r_1} \not\equiv g^{r_2} \pmod{p}$ . (We implicitly assume that coins  $r_1$  and  $r_2$  are selected from the coin space  $\mathbb{Z}_q^*$ , hence,  $g^{r_1} = g^{r_1} \pmod{p}$  and  $g^{r_2} = g^{r_2} \pmod{p}$ .) It follows that  $c_1 \neq c_2$ . Otherwise ( $r_1 = r_2$ ), we have  $m_1 \neq m_2$  and we proceed as follows. By definition of  $\text{Enc}$ , we have  $c_1[2] = h^{r_1} \cdot m_1 \pmod{p}$  and  $c_2[2] = h^{r_2} \cdot m_2 \pmod{p}$ . Since  $h \neq 0$ , we have  $h^{r_1} \cdot m_1 \not\equiv h^{r_1} \cdot m_2 \pmod{p}$ .  $\square$

Given that ciphertexts generated by the modified version of El Gamal used in JCJ [79, §4] encapsulate El Gamal ciphertexts, the proof of key construction by Gennaro et al. [57] is sufficient to ensure that El Gamal is collision-free:

**Corollary 25.** *The modified version of El Gamal used in JCJ [79, §4] is collision-free its message space  $\mathfrak{m}$ .*

The sigma protocol for proving correct key construction by Gennaro et al. [57] does not explicitly require the suitability of cryptographic parameters to be checked, hence, Lemma 24 is not immediately applicable. Nonetheless, we can trivially make the necessary checks explicit and, hence, the non-interactive proof system derived from the sigma protocol for proving correct key construction by Gennaro et al. is sufficient to ensure that El Gamal is collision-free. It follows that JCJ satisfies Injectivity, hence, JCJ is an election scheme.

## APPENDIX H PROOF: JCJ IS VERIFIABLE

Elections schemes constructed from generalized JCJ satisfy individual (§H-A), universal (§H-B) and eligibility (§H-C) verifiability, hence, such schemes satisfy election verifiability

with internal authentication (§H-D). It follows that JCJ satisfies election verifiability (§H-E).

### A. Individual verifiability

**Proposition 26.** *Suppose  $\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6$  and  $\mathcal{H}$  satisfy the preconditions of Figure 3. Further suppose that  $\Gamma$  is collision-free for its message space  $\mathfrak{m}$  and  $\Sigma_1$  satisfies special soundness and special honest verifier zero-knowledge. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies individual verifiability.*

*Proof.* Let  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$  and  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ . Suppose  $k$  is a security parameter,  $PK_{\mathcal{T}}$  is a public key,  $n_C$  is an integer, and  $\beta$  and  $\beta'$  are choices. Further suppose that  $(pk, sk)$  and  $(pk', sk')$  are key pairs and  $b$  and  $b'$  are ballots such that  $(pk, sk) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ ,  $(pk', sk') \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ ,  $b \leftarrow \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$ ,  $b' \leftarrow \text{Vote}(sk', PK_{\mathcal{T}}, n_C, \beta', k)$ ,  $b \neq \perp$ , and  $b' \neq \perp$ . By definition of  $\text{Vote}$ , we have  $PK_{\mathcal{T}}$  is a vector  $(pk_T, \mathfrak{m}, \rho)$  and  $\text{VerKey}((1^k, pk_T, \mathfrak{m}), \rho) = 1$ . By definition of  $\text{Vote}$ ,  $b[2]$  and  $b'[2]$  are ciphertexts such that  $b[2] \leftarrow \text{Enc}(pk_T, sk)$  and  $b'[2] \leftarrow \text{Enc}(pk_T, sk')$ , where  $sk, sk' \in \mathfrak{m}$ . Furthermore, the ciphertexts are constructed using random coins—i.e., the coins used by  $b[2]$  and  $b'[2]$  will be distinct with overwhelming probability. Since  $\Gamma$  is collision-free for  $\mathfrak{m}$ , we have  $b[2] \neq b'[2]$  and  $b \neq b'$  with overwhelming probability, concluding our proof.  $\square$

### B. Universal verifiability.

**Proposition 27.** *Suppose  $\Gamma$  is a homomorphic asymmetric encryption scheme,  $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5$  and  $\Sigma_6$ , are sigma protocols and  $\mathcal{H}$  is a hash function such that the conditions of Figure 3 are satisfied. Further suppose that  $\Gamma$  satisfies IND-CPA and  $\Sigma_1$  and  $\Sigma_6$  satisfy special soundness and special honest verifier zero-knowledge. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies universal verifiability.*

The proof is similar in structure to the universal verifiability proof for Helios (§D-B): we use the definition of the verification algorithm to construct the tally  $\mathbf{X}$  given by the adversary, and then show that  $\mathbf{X}$  is equal to the correct tally.

*Proof.* Suppose that an execution of  $\text{Exp-UV-Int}(\Pi, \mathcal{A}, k)$  computes

```

(PKT, nV) ← A(k);
for 1 ≤ i ≤ nV do (pki, ski) ← Register(PKT, k)
L ← {pk1, ..., pknV};
M ← {(pk1, sk1), ..., (pknV, sknV)};
(BB, nC, X, P) ← A(M);
Y ← correct-tally(PKT, BB, M, nC, k);

```

such that  $\text{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$ . The JCJ verification algorithm checks the proof  $\rho$  in  $PK_{\mathcal{T}} = (pk_T, \mathfrak{m}, \rho)$ , so  $\text{VerKey}((1^k, pk_T, \mathfrak{m}), \rho) = 1$  and by simulation sound

extractability we are assured that  $pk_T$  was honestly generated, i.e., there exists  $r$  and  $SK_T$  such that  $(pk_T, SK_T, m) = \text{Gen}(1^k; r)$ . We now look at each step in the Verify algorithm.

- *Check removal of invalid ballots:* Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $BB$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a vector of length 4 and  $\text{VerCiph}((pk_T, b_i[1]\{1, \dots, n_C\}), b_i[3]) = 1 \wedge \text{VerBind}((pk_T, b_i[1], b_i[2]), b_i[4]) = 1$ . If this set is empty, then Verify would only accept if  $\mathbf{X}[i] = 0$  for all  $1 \leq i \leq n_C$  and  $\mathbf{P} = \perp$ . Since the set is empty, no ballots  $b$  were posted to the bulletin board for which  $\text{VerCiph}((pk_T, b_i[1], \{1, \dots, n_C\}), b_i[3]) = 1 \wedge \text{VerBind}((pk_T, b_i[1], b_i[2]), b_i[4]) = 1$ . By the completeness of the zero knowledge proofs, if the ballots were outputs of the Vote function, then they would verify. Therefore, no ballots on the bulletin board were the output of the Vote function, so we will have that  $\mathbf{Y}$  is also a vector of zeroes. Thus we would have that  $\mathbf{X} = \mathbf{Y}$  and conclude our proof. Now let's assume that  $\{b_1, \dots, b_\ell\} \neq \emptyset$ .

We must have for all choices  $\beta \in \{1, \dots, n_C\}$ , secret keys  $sk$  such that  $(pk, sk) \in M$ , coins  $r$ , and ballots  $b = \text{Vote}(sk, PK_T, n_C, \beta, k; r)$  that  $b \notin BB \setminus \{b_1, \dots, b_\ell\}$  with overwhelming probability, since otherwise we would have a contradiction:  $\{b_1, \dots, b_\ell\}$  is not the largest subset of  $BB$  satisfying the conditions of the Tally algorithm. Therefore, we must have that

$$\begin{aligned} & \text{correct-tally}(PK_T, M, BB, n_C, k) \\ &= \text{correct-tally}(PK_T, M, \{b_1, \dots, b_\ell\}, n_C, k) \end{aligned} \quad (5)$$

- *Check duplicate elimination:* Next, the verification algorithm checks that duplicate votes were properly eliminated, i.e., that either  $|\mathbf{P}_{\text{dupl}}[i]| = 1 \wedge \exists j \in \{1, \dots, i-1, i+1, \dots, \ell\}$  such that  $\text{VerPET}((pk_T, b_i[2], b_j[2]), \mathbf{P}_{\text{dupl}}[i][1], 1) = 1$  or  $|\mathbf{P}_{\text{dupl}}[i]| = \ell - 1 \wedge \forall j \in \{1, \dots, i-1, i+1, \dots, n\}$  such that  $\text{VerPET}((pk_T, b_i[2], b_j[2]), 0, \mathbf{P}_{\text{dupl}}[i][j]) = 1$ . Let  $\mathbf{BB}$  be constructed as in Step (2) of the JCJ tallying algorithm. By the simulation sound extractability of the  $\mathbf{P}_{\text{dupl}}[i]$ , we are assured that there are no duplicate votes in  $\mathbf{BB}$ .
- *Check mixing:* Now the ballots in  $\mathbf{BB}$  are permuted and re-encrypted using a mixnet. While permuting the ballots isn't necessary for verifiability, the associated proofs are necessary because they show that the re-encryption was done properly (for example, they ensure that the encrypted ballot was multiplied by an encryption of the identity element, and not some other group element that might change the vote). Let  $C_1$  denote the list of mixed re-encryptions of candidates,  $C_2$  denote the list of mixed re-encryptions of voters' secret keys from the ballots, and  $C_3$  denote the mixed list of encryptions of voters' secret keys. The permutation used to generate  $C_3$  is different from the permutation used to generate  $C_1$  and  $C_2$ , but this isn't important to the verifiability of the scheme. We have

that  $\text{VerMix}((pk_T, (b'_1[1], \dots, b'_{|\mathbf{BB}|}[1]), \mathbf{C}_1), P_{\text{mix},1}) = 1 \wedge \text{VerMix}((pk_T, (b'_1[2], \dots, b'_{|\mathbf{BB}|}[2]), \mathbf{C}_2), P_{\text{mix},2}) = 1 \wedge \text{VerMix}((pk_T, L, \mathbf{C}_3), P_{\text{mix},3}) = 1$ . By simulation sound extractability, we have that each  $C_i$  does indeed contain re-encryptions of the original lists in  $\mathbf{BB}$ .

- *Check removal of ineligible ballots:* Next, the verification algorithm ensures that ineligible ballots are removed properly. The verification algorithm checks that each PET in  $\mathbf{P}[8] = \mathbf{P}_{\text{inelig}}$  is valid. Let  $C'_1 \subseteq C_1$  be the set of  $C_1[i] \in C_1$  for which  $|\mathbf{P}_{\text{inelig}}| = 1$  and there exists  $c \in C_3$  such that  $\text{VerPET}((pk_T, \mathbf{C}_2[i], c, 1), \mathbf{P}_{\text{inelig}}[i][1]) = 1$ . In other words,  $C'_1$  is the set of encryptions of candidates generated using a valid voter's secret key.
- *Check decryption:* Finally, the verification algorithm checks the proofs that all of the ballots in  $C'_1$  are properly decrypted. The verification algorithm outputs 1, so by simulation sound extractability we are assured that the multiset of candidates given by decrypting the ballots in  $C'_1$  is correct. We will call this multiset  $C_{\text{Final}}$ . Finally the verification algorithm checks that this multiset corresponds to the vector  $\mathbf{X}$ .

We can see that  $C_{\text{Final}}$  satisfies the following properties. First, every element  $\beta$  in  $C_{\text{Final}}$  corresponds to a ballot  $b \in BB$  which was generated using Vote with a valid voter's secret key. This is ensured by steps (1), (3), and (4) of the verification algorithm. Second, for every  $\beta \in C_{\text{Final}}$ , the ballot corresponding to this  $\beta$  was the only one constructed under its particular secret key, i.e., (where  $b$  is the ballot corresponding to  $\beta$ )  $\neg \exists b', \beta', r' : b' \in BB \setminus \{b\} \wedge b' = \text{Vote}(sk, PK_T, n_C, \beta', k; r')$ . This is ensured by steps (2) and (3) of the verification algorithm. Therefore, we have that each  $\beta \in C_{\text{Final}}$  corresponds to a ballot in  $\text{authorized}(PK_T, BB, M, n_C, k)$ . Finally  $\mathbf{X}[\beta] = k$  iff  $\exists \beta^k \in C_{\text{Final}}$ . This is ensured by step (5) of the verification algorithm. Since these are the exact properties that define  $\text{correct-tally}(PK_T, M, \{b_1, \dots, b_\ell\}, n_C, k)$ , we must have that  $\mathbf{X} = \mathbf{Y}$ .  $\square$

### C. Eligibility Verifiability

We proceed as follows. First, we derive an IND-1-CPA encryption scheme from generalized JCJ (§H-C1). Secondly, we introduce an experiment that is equivalent to Exp-EV-Int-Weak for JCJ (§H-C2). Finally, we prove that JCJ satisfies our new experiment (§H-C3), using the IND-1-CPA encryption scheme.

#### 1) Encryption scheme from generalized JCJ:

**Definition 30.** Suppose  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is an asymmetric encryption scheme,  $\Sigma_1$  proves correct key construction,  $\Sigma_3$  proves conjunctive plaintext knowledge, and  $\mathcal{H}$  is a random oracle. Let  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$  and  $\text{FS}(\Sigma_3, \mathcal{H}) = (\text{ProveBind}, \text{VerBind})$ . We define  $\Pi_{\text{JCJ}}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H}) = (\text{Gen}', \text{Enc}', \text{Dec}')$  as follows:

- $\text{Gen}'(1^k; r) : \text{Compute } (pk_T, SK_T, m) \leftarrow \text{Gen}(1^k; r); \rho \leftarrow \text{ProveKey}((1^k, PK_T, m), (SK_T, r)); PK_T \leftarrow$

- $(pk_T, m, \rho)$ ;  $m' \leftarrow \{(m_1, m_2) \mid m_1, m_2 \in \mathfrak{m}\}$ . *Output*  $((PK_{\mathcal{T}}, k), SK_{\mathcal{T}}, m')$ .
- $\text{Enc}'(pk, m)$  : Parse  $m$  as a vector  $(\beta, d)$ ,  $pk$  as a vector  $(PK_{\mathcal{T}}, k)$ , and  $PK_{\mathcal{T}}$  as a vector  $(pk_T, m, \rho)$ , outputting  $\perp$  if parsing fails. Select coins  $r_1$  and  $r_2$  and compute  $c_1 \leftarrow \text{Enc}(pk_T, \beta; r_1)$ ;  $c_2 \leftarrow \text{Enc}(pk_T, d; r_2)$ ;  $\tau \leftarrow \text{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, d, r_2))$ . *Output*  $(c_1, c_2, \tau)$ .
  - $\text{Dec}'(pk, sk, c)$  : Parse  $c$  as  $(c_1, c_2, \tau)$ ,  $pk$  as  $(PK_{\mathcal{T}}, k)$ , and  $PK_{\mathcal{T}}$  as  $(pk_T, m, \rho)$ , outputting  $\perp$  if parsing fails or  $\text{VerBind}((pk_T, c_1, c_2), \tau) \neq 1$ . Compute  $\beta \leftarrow \text{Dec}(pk_T, sk, c_1)$ ;  $d \leftarrow \text{Dec}(pk_T, sk, c_2)$  and output  $(\beta, d)$ .

The key generation algorithm  $\text{Gen}'$  outputs a public key  $(PK_{\mathcal{T}}, k)$ , where  $PK_{\mathcal{T}} = (pk_T, m, \rho)$ . Parameters  $m, \rho$ , and  $k$  are used in our proof of eligibility verifiability, but are not required by the encryption scheme.

**Proposition 28.**  $\Pi_{J CJ}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H})$  is an asymmetric encryption scheme satisfying IND-1-CPA, where  $\Pi, \Sigma_1, \Sigma_3$  and  $\mathcal{H}$  satisfy the preconditions of Definition 30.

*Proof.* The proof that this scheme satisfies IND-1-CPA is adapted from that of [21, Theorem 5.1]. We will show that if there is an adversary  $\mathcal{A}'$  that can win the IND-1-CPA game for the scheme, then there is another adversary  $\mathcal{A}$  that can win the IND-CPA game for the following scheme: Let  $\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})$  be an asymmetric encryption scheme satisfying IND-CPA. Define  $\Gamma' = (\text{Gen}', \text{Enc}', \text{Dec}')$  as follows:

- $\text{Gen}'(1^k; r)$  : Compute  $(pk, sk, m) \leftarrow \text{Gen}(1^k; r)$ ;  $\rho \leftarrow \text{ProveKey}((1^k, pk, m), (sk, r))$ ;  $PK_{\mathcal{T}} \leftarrow (pk, m, \rho)$ ;  $pk' = (PK_{\mathcal{T}}, k)$ ;  $m' \leftarrow \{(m_1, m_2) \mid m_1, m_2 \in \mathfrak{m}\}$ , and output  $(pk', sk, m')$ .
- $\text{Enc}'(pk, m)$  : Parse  $m$  as a vector  $(m_0, m_1)$ ,  $pk$  as  $(PK_{\mathcal{T}}, k)$ , and  $PK_{\mathcal{T}}$  as  $(pk', m, \rho)$ , outputting  $\perp$  if parsing fails. Compute  $c_0 \leftarrow \text{Enc}(pk', m_0)$ ;  $c_1 \leftarrow \text{Enc}(pk', m_1)$ , and output  $(c_0, c_1)$ .
- $\text{Dec}'(pk, sk, c)$  : Parse  $c$  as a vector  $(c_0, c_1)$ ,  $pk$  as  $(PK_{\mathcal{T}}, k)$ , and  $PK_{\mathcal{T}}$  as  $(pk', m, \rho)$ , outputting  $\perp$  if parsing fails. Compute  $m_0 \leftarrow \text{Dec}(pk', sk, c_0)$ ;  $m_1 \leftarrow \text{Dec}(pk', sk, c_1)$ , and output  $(m_0, m_1)$ .

It is straightforward to see that this scheme satisfies IND-CPA.

Now we begin the reduction. Let  $\mathcal{A}'$  be an adversary that wins the IND-1-CPA game against  $\Pi_{J CJ}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H})$  with non-negligible probability. We will construct an adversary  $\mathcal{A}$  that wins the IND-CPA game against the  $\Gamma'$  defined above with non-negligible probability.  $\mathcal{A}$  is first given a public key  $pk$ , where  $pk = (PK_{\mathcal{T}}, k)$  and  $PK_{\mathcal{T}} = (pk', m, \rho)$ .  $\mathcal{A}$  forwards  $pk$  to  $\mathcal{A}'$ .  $\mathcal{A}'$  may make queries to its random oracle.  $\mathcal{A}$  will simulate the random oracle and keep a list  $\mathcal{H}$  of all previously asked queries. If  $\mathcal{A}'$  makes a query for a value already in  $\mathcal{H}$ ,  $\mathcal{A}$  responds with a value consistent with the list. If  $\mathcal{A}'$  makes a query for a new value,  $\mathcal{A}$  chooses a value uniformly at random from the range of the random oracle and adds the query/response pair to  $\mathcal{H}$ . We will denote by  $\mathcal{H}(x)$  the response  $y$  such that  $(x, y)$  is in  $\mathcal{H}$ , and  $\perp$  if no such

query/response pair is in  $\mathcal{H}$ .

Next  $\mathcal{A}'$  will output two messages  $m_0, m_1$  of the form  $(\beta, d)$ .  $\mathcal{A}$  outputs  $m_0, m_1$  and receives a challenge ciphertext  $c^* = (c_0^*, c_1^*)$ .  $\mathcal{A}$  then picks a challenge  $chal^*$  at random from the challenge space. In order to generate the proof of conjunctive plaintext knowledge that  $\mathcal{A}'$  expects,  $\mathcal{A}$  will use the simulator  $Sim$  for the sigma protocol associated with  $\text{ProveBind}$ . This simulator exists due to the special honest verifier zero knowledge property of the sigma protocol.  $\mathcal{A}$  runs  $Sim((pk', c_0^*, c_1^*), chal^*)$  to obtain the simulated proof  $\tau^* = (comm^*, resp^*)$ , and adds the pair  $((pk', c_0^*, c_1^*) || comm^*, chal^*)$  to  $\mathcal{H}$ . If there is already an entry corresponding to the query  $(pk', c_0^*, c_1^*) || comm^*$  in  $\mathcal{H}$ ,  $\mathcal{A}$  aborts with “Error 1”.  $\mathcal{A}$  then gives  $(c_0^*, c_1^*, \tau^*)$  to  $\mathcal{A}'$ .

$\mathcal{A}'$  will next output its vector of decryption queries  $c$ . Let  $|c| = \ell$ . For each  $i \in \{1, \dots, \ell\}$ ,  $\mathcal{A}$  will obtain the response to the query  $c[i]$  using the following procedure. First,  $\mathcal{A}$  checks that  $c[i]$  is a valid ciphertext, i.e., that  $c[i] = (c_i^0, c_i^1, \tau_i)$  where  $\tau_i = (comm_i, resp_i)$  such that  $\text{VerBind}((pk', c_i^0, c_i^1), (comm_i, \mathcal{H}((pk', c_i^0, c_i^1) || comm_i)), resp_i) = 1$ . If there is no entry  $(x, y) \in \mathcal{H}$  such that  $x = (pk', c_i^0, c_i^1) || comm_i$ ,  $\mathcal{A}$  adds it as if  $\mathcal{A}'$  had queried its random oracle on that value. If these conditions do not hold or  $c[i] = (c_0^*, c_1^*, \tau^*)$ , the response for  $c[i]$  will be  $\perp$ . Now  $\mathcal{A}$  checks to see where  $\mathcal{A}'$  queried on  $(pk', c_i^0, c_i^1) || comm_i$ . If  $\mathcal{A}'$  never made such a query,  $\mathcal{A}$  aborts with “Error 2”.  $\mathcal{A}$  simulates a new copy of  $\mathcal{A}'$  up to the point of that query, but this time responds with a new, uniformly random value. All other queries are answered as they were in the “main” run of  $\mathcal{A}'$ .  $\mathcal{A}$  continues the simulation until  $\mathcal{A}'$  outputs  $c'$ . If  $c'$  contains an entry  $(c_j^0, c_j^1, \tau_j)$  such that  $c_j^0 = c_i^0, c_j^1 = c_i^1$  and  $comm_j = comm_i$ , then  $\mathcal{A}$  uses the special soundness extractor for the sigma protocol to obtain the witness  $w_i$  for the statement. This witness consists of the messages and random coins used to generate the ciphertexts.  $\mathcal{A}$  uses this witness to answer the decryption query in the “main” run. Finally,  $\mathcal{A}'$  will output a bit  $b$ , which  $\mathcal{A}$  outputs as well.

The remainder of the proof is almost exactly the same as that of [21, Theorem 5.1], and so is omitted here.  $\square$

## 2) Variant of Exp-EV-Int-Weak:

$\text{Exp-EV-1-Int}(\Pi, \mathcal{A}, k) =$

- 1  $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k)$ ;
- 2  $(pk, sk) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ ;
- 3  $Rvld \leftarrow \emptyset$ ;
- 4  $(n_C, \beta, b) \leftarrow \mathcal{A}^R(PK_{\mathcal{T}}, pk, k)$ ;
- 5 **if**  $\exists r : b = \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp \wedge b \notin Rvld$  **then**
- 6     **return** 1
- 7 **else**
- 8     **return** 0

**Lemma 29.** Let  $\Pi$  be Generalized J CJ, where the encryption

scheme  $\Gamma$  satisfies IND-CPA. Then we have

$$\begin{aligned} & \forall \mathcal{A} \exists \mu \forall k . \text{Succ}(\text{Exp-EV-1-Int}(\Pi, \mathcal{A}, k)) \leq \mu(k) \\ \Leftrightarrow & \forall \mathcal{A}' \exists \mu' \forall k' . \text{Succ}(\text{Exp-EV-1-Int-Weak}(\Pi, \mathcal{A}', k')) \leq \mu'(k'), \end{aligned}$$

where  $\mathcal{A}$  and  $\mathcal{A}'$  are PPT adversaries,  $\mu$  and  $\mu'$  are negligible functions, and  $k$  and  $k'$  are security parameters.

The forward implication is required by Proposition 30 and we provide a formal proof below. A proof of the reverse implication is straight-forward and we omit our formal proof.

*Proof.* We will show that if an adversary wins Exp-EV-1-Int-Weak, then there exists an adversary that wins Exp-EV-1-Int. Let  $\mathcal{A}'$  be the adversary that wins Exp-EV-1-Int-Weak with non-negligible probability. We will construct the adversary  $\mathcal{A}$  for Exp-EV-1-Int. The challenger first computes  $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \text{Setup}(k)$  and  $(pk, sk) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ .  $\mathcal{A}$  is given as input  $(PK_{\mathcal{T}}, pk, k)$  and forwards  $(PK_{\mathcal{T}}, k)$  to  $\mathcal{A}'$ .  $\mathcal{A}'$  outputs  $n_V$ .

Now  $\mathcal{A}'$  may make some oracle queries.  $\mathcal{A}$  will maintain a list  $H$  of  $(i, pk'_i, sk'_i)$  tuples.  $\mathcal{A}'$ 's first oracle,  $C$ , needs to return secret keys associated with the  $pk_i$ . When  $\mathcal{A}$  receives a query  $C(i)$ ,  $\mathcal{A}$  checks if  $(i, pk'_i, sk'_i) \in H$ . If so,  $\mathcal{A}$  returns  $sk'_i$ . Otherwise,  $\mathcal{A}$  computes  $(pk'_i, sk'_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ , adds  $(i, pk'_i, sk'_i)$  to  $H$ , and returns  $sk'_i$ . Again by the IND-CPA property of the encryption scheme,  $\mathcal{A}'$  cannot tell that  $sk'_i$  does not actually correspond to  $pk_i$ .  $\mathcal{A}'$ 's second oracle,  $R$  can be queried on inputs  $(i, \beta, n_C)$ , on which it returns  $\text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k)$ . If  $\mathcal{A}$  receives the query  $R(i, \beta, n_C)$ , it checks if  $(i, pk'_i, sk'_i) \in H$ . If so,  $\mathcal{A}$  computes  $b = \text{Vote}(sk'_i, PK_{\mathcal{T}}, n_C, \beta, k)$  and returns  $b$ . Otherwise,  $\mathcal{A}$  computes  $(pk'_i, sk'_i) \leftarrow \text{Register}(PK_{\mathcal{T}}, k)$ , adds  $(i, pk'_i, sk'_i)$  to  $H$ , then computes  $b = \text{Vote}(sk'_i, PK_{\mathcal{T}}, n_C, \beta, k)$  and returns  $b$ . Again by the IND-CPA property of the encryption scheme,  $\mathcal{A}'$  cannot tell that the ballots  $b$  he receives were computed with a secret key that does not correspond to  $pk_i$ . Finally,  $\mathcal{A}'$  outputs  $(n_C, \beta, i, b)$ , and  $\mathcal{A}$  outputs  $(n_C, \beta, b)$ . Clearly,  $\mathcal{A}$  has the same success probability as  $\mathcal{A}'$ , so  $\mathcal{A}$  wins Exp-EV-1-Int with non-negligible probability.  $\square$

### 3) Eligibility Verifiability:

**Proposition 30.** *Suppose  $\Gamma$  is a homomorphic asymmetric encryption scheme,  $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5$  and  $\Sigma_6$ , are sigma protocols and  $\mathcal{H}$  is a hash function such that the conditions of Figure 3 are satisfied. Further suppose that  $\Gamma$  satisfies IND-CPA. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies eligibility verifiability.*

*Proof.* Let  $\Pi_{\text{JCJ}} = (\text{Gen}', \text{Enc}', \text{Dec}')$  be defined as above. Let  $\mathcal{A}'$  be an adversary that wins the Exp-EV-1-Int game. We will construct the adversary  $\mathcal{A}$  that wins the IND-1-CPA game with non-negligible advantage. The challenger first generates  $(PK, SK_{\mathcal{T}}, m) \leftarrow \text{Gen}'(k)$ , where  $PK = (PK_{\mathcal{T}}, k)$  and  $PK_{\mathcal{T}} = (pk_{\mathcal{T}}, m, \rho)$ , and gives  $(PK_{\mathcal{T}}, k)$  to  $\mathcal{A}$  as input.  $\mathcal{A}$  runs  $\text{Register}(PK_{\mathcal{T}}, k)$  twice to get  $(pk_0, sk_0), (pk_1, sk_1)$  and sets  $m_0 = (1, sk_0), m_1 = (1, sk_1)$ .  $\mathcal{A}$  then outputs  $(m_0, m_1)$ . The challenger picks a bit  $b$  at random and gives

$c = \text{Enc}'(PK, m_b)$  to  $\mathcal{A}$ . We have  $c = (c_1, c_2, \tau)$ , where  $c_2 = \text{Enc}(pk_{\mathcal{T}}, sk_b)$ . Now  $\mathcal{A}$  begins to interact with  $\mathcal{A}'$  by giving  $(PK_{\mathcal{T}}, c_2, k)$  to  $\mathcal{A}'$ .

At this point  $\mathcal{A}'$  may call its oracle  $R$ . If  $\mathcal{A}$  receives a query  $R(\beta, n_C)$ , it will construct  $x \leftarrow \text{Vote}(sk_0, PK_{\mathcal{T}}, n_C, \beta, k)$  and return  $x$ . We have that  $x = (\text{Enc}(pk_{\mathcal{T}}, \beta), \text{Enc}(pk_{\mathcal{T}}, sk_0), \sigma, \tau)$ , where  $\sigma$  and  $\tau$  are proofs of plaintext knowledge in a subspace and conjunctive plaintext knowledge, respectively. By the IND-CPA property of  $\text{Enc}$ ,  $\mathcal{A}'$  can't distinguish between encryptions of  $sk_0$  and  $sk_1$ . Therefore we can construct  $x$  using  $sk_0$  even if the secret key corresponding to  $c_2$  is actually  $sk_1$ .

$\mathcal{A}'$  will then output  $(n_C, \beta, b^*)$ , where  $b^* = (c_1^*, c_2^*, \sigma^*, \tau^*)$ .  $\mathcal{A}'$  wins with probability  $\frac{1}{p(k)}$  for some polynomial  $p$ , so with probability  $\frac{1}{p(k)}$  we have that  $c_1 = \text{Enc}(pk_{\mathcal{T}}, \beta; r_1), c_2 = \text{Enc}(pk_{\mathcal{T}}, sk_b; r_2), \sigma = \text{ProveCiph}((pk_{\mathcal{T}}, c_1, \{1, \dots, n_C\}), (\beta, r_1))$ , and  $\tau^* = \text{ProveBind}((pk_{\mathcal{T}}, c_1, c_2), (\beta, r_1, sk_b, r_2))$ . In order to ensure that we get a ballot of this form from  $\mathcal{A}'$  with high enough probability,  $\mathcal{A}$  repeats the above interaction with  $\mathcal{A}'$   $p(k)$  times to obtain  $(n_C^1, \beta^1, b_1^*), \dots, (n_C^{p(k)}, \beta^{p(k)}, b_{p(k)}^*)$ .  $\mathcal{A}$  outputs  $(b_1^*[1], b_1^*[2], b_1^*[4]), \dots, (b_{p(k)}^*[1], b_{p(k)}^*[2], b_{p(k)}^*[4])$ , and receives  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_1^*[1], b_1^*[2], b_1^*[4]), \dots, \text{Dec}'(PK, SK_{\mathcal{T}}, (b_{p(k)}^*[1], b_{p(k)}^*[2], b_{p(k)}^*[4]))$ . If there exist  $i, j$  such that  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^i, sk_0)$  and  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_j^*[1], b_j^*[2], b_j^*[4])) = (\beta^j, sk_1)$ , or there exists no  $i$  such that  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^i, sk_0)$  or  $(\beta^i, sk_1)$ , then  $\mathcal{A}$  outputs a random bit. Otherwise, if there exists  $i$  such that  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^i, sk_0)$ , then  $\mathcal{A}$  outputs 0. Likewise, if there exists  $i$  such that  $\text{Dec}'(PK, SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^i, sk_1)$ , then  $\mathcal{A}$  outputs 1.

We now argue that  $\mathcal{A}$  can determine the correct bit  $b$  with non-negligible advantage.

There are three possible events that can occur in a run of  $\mathcal{A}$ . The first possibility is that  $\mathcal{A}'$  fails on each of its  $p(k)$  runs so that  $\mathcal{A}$  has to guess. This occurs with probability  $(1 - \frac{1}{p(k)})^{p(k)}$ . The second event is that  $\mathcal{A}'$  does succeed in one of its runs, but on a different run it outputs

$$\begin{aligned} b = & (\text{Enc}(PK_{\mathcal{T}}, \beta; r_1), \\ & \text{Enc}(PK_{\mathcal{T}}, sk_{(1-b)}; r_2), \\ & \text{ProveCiph}((PK_{\mathcal{T}}, c_1, \{1, \dots, n_C\}), (\beta, r_1)), \\ & \text{ProveBind}((PK_{\mathcal{T}}, c_1, c_2), (\beta, r_1, sk_{(1-b)}, r_2))). \end{aligned}$$

However, because  $sk_0$  and  $sk_1$  are chosen randomly, the probability of this occurring is negligible. Finally, the third possibility is that  $\mathcal{A}'$  succeeds in at least one of its runs. This occurs with probability  $\sum_{i=0}^{p(k)-1} (1 - \frac{1}{p(k)})^i (\frac{1}{p(k)})$ . In the first two events,  $\mathcal{A}$  guesses and wins with probability  $\frac{1}{2}$ , and in the third event  $\mathcal{A}$  wins with probability 1. Therefore, the total probability that  $\mathcal{A}$  wins is  $(\sum_{i=0}^{p(k)-1} (1 - \frac{1}{p(k)})^i (\frac{1}{p(k)})) + \frac{1}{2} (1 - \frac{1}{p(k)})^{p(k)} + \frac{1}{2} \mu(k)$ , for some negligible function  $\mu$ .

We have that this equation is equal to:

$$\begin{aligned}
&= \frac{1}{p(k)} \sum_{i=0}^{p(k)-1} \left(1 - \frac{1}{p(k)}\right)^i + \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} \\
&\quad + \frac{1}{2} \mu(k) \\
&= \frac{1}{p(k)} \left(p(k) - \left(1 - \frac{1}{p(k)}\right)^{p(k)} p(k)\right) + \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} \\
&\quad + \frac{1}{2} \mu(k) \\
&= 1 - \left(1 - \frac{1}{p(k)}\right)^{p(k)} + \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} + \frac{1}{2} \mu(k) \\
&= 1 - \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} + \frac{1}{2} \mu(k)
\end{aligned}$$

In order to determine the advantage of this adversary, we subtract  $\frac{1}{2}$  from this:

$$\begin{aligned}
&1 - \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} + \frac{1}{2} \mu(k) - \frac{1}{2} \\
&= \frac{1}{2} - \frac{1}{2} \left(1 - \frac{1}{p(k)}\right)^{p(k)} + \frac{1}{2} \mu(k)
\end{aligned}$$

As  $k$  gets large,  $\left(1 - \frac{1}{p(k)}\right)^{p(k)}$  converges to  $\frac{1}{e}$  and  $\mu(k)$  goes to 0, so the entire equation converges to  $\frac{1}{2} - \frac{1}{2e}$ . This is non-negligible.

Combining this reduction with Lemma 29, we have that if  $\Pi_{JCJ}$  satisfies IND-1-CPA, then JCJ satisfies Exp-EV-Int-Weak.  $\square$

#### D. Election verifiability

By Propositions 26, 27, & 30, election schemes constructed from generalized JCJ satisfy election verifiability with internal authentication:

**Corollary 31.** *Suppose  $\Gamma$ ,  $\Sigma_1$ ,  $\Sigma_2$ ,  $\Sigma_3$ ,  $\Sigma_4$ ,  $\Sigma_5$ ,  $\Sigma_6$  and  $\mathcal{H}$  satisfy the preconditions of Figure 3. Further suppose that  $\Gamma$  satisfies IND-CPA and is collision-free,  $\Sigma_1$  and  $\Sigma_6$  satisfy special soundness and special honest verifier zero-knowledge, and  $\mathcal{H}$  is a random oracle. We have  $\text{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$  satisfies election verifiability with internal authentication.*

#### E. Proof: Theorem 5

*Proof of Theorem 5.* We know that  $\Gamma$  satisfies IND-CPA, and by Corollary 25  $\Gamma$  is also collision-free. Therefore the proof follows from Corollary 31, subject to the applicability of Theorem 9 to the mixnet and sigma protocol used by JCJ to prove correct key construction.  $\square$

### APPENDIX I JUELS ET AL. DEFINITIONS

Juels et al. [79, §2] define an election scheme as a tuple of (Register, Vote, Tally, Verify) probabilistic polynomial-time algorithms:

- **Register**, denoted  $(pk, sk) \leftarrow \text{Register}(SK_{\mathcal{R}}, i, k_1)$ , is executed by the registrars. Register takes as input the private key  $SK_{\mathcal{R}}$  of the registrars, a voter's identity  $i$ , and security parameter  $k_1$ . It outputs a credential pair  $(pk, sk)$ .
- **Vote**, denoted  $b \leftarrow \text{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k_2)$ , is executed by voters. Vote takes as input a voter's private credential  $sk$ , the public key  $PK_{\mathcal{T}}$  of the tallier, the number of candidates  $n_C$ , the voter's choice  $\beta$ , and security parameter  $k_2$ . It outputs a ballot  $b$ .
- **Tally**, denoted  $(\mathbf{X}, P) \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$ , is executed by the tallier. Tally takes as input the private key  $SK_{\mathcal{T}}$  of the tallier, the bulletin board  $BB$ , the number of candidates  $n_C$ , the set containing voters'

public credentials, and security parameter  $k_3$ . It outputs the tally  $\mathbf{X}$  and a proof  $P$  that the tally is correct.

- **Verify**, denoted  $v \leftarrow \text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P)$ , can be executed by anyone to audit the election. Verify takes as input the public key  $PK_{\mathcal{R}}$  of the registrars, the public key  $PK_{\mathcal{T}}$  of the tallier, the bulletin board  $BB$ , the number of candidates  $n_C$ , and a candidate proof  $P$  of correct tallying. It outputs a bit  $v$ , which is 1 if the tally successfully verifies and 0 on failure.

The above definition fixes an apparent oversight in JCJ's presentation: we supply the registrars' public key as input to the verification algorithm, because that key would be required by Verify to check the signature on the electoral roll.

Juels et al. [79, §3] formalize *correctness* and *verifiability* to capture their notion of election verifiability. We rename those to *JCJ-correctness* and *JCJ-verifiability* to avoid ambiguity. For readability, the definitions we give below contain subtle differences from the original presentation. For example, we sometimes use for loops instead of pattern matching.

JCJ-correctness asserts that an adversary cannot modify or eliminate votes of honest voters, and stipulates that at most one ballot is tallied per voter. Intuitively, the security definition challenges the adversary to ensure that verification succeeds and the tally<sup>35</sup> does not include some honest votes or contains too many votes. The definition of JCJ-correctness fixes apparent errors in the original presentation: the adversary is given the credentials for corrupt voters and distinct security parameters are supplied to the Register and Vote algorithms. An implicit assumption is also omitted:  $\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'}$  is a multiset of valid votes, that is, for all  $\beta \in \{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'}$  we have  $1 \leq \beta \leq n_C$ . Without this assumption the security definition cannot be satisfied by many election schemes, including the election scheme by Juels et al.

**Definition 31** (JCJ-correctness). *An election scheme  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  satisfies JCJ-correctness if for all probabilistic polynomial-time adversary  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all positive integers  $n_C$  and  $n_V$ , and security parameters  $k_1, k_2$ , and  $k_3$ , we have  $\text{Succ}(\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \leq \mu(k_1, k_2, k_3)$ , where Exp-JCJ-Cor is defined as follows:<sup>36</sup>*

$$\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3) =$$

<sup>35</sup> Juels et al. translate tallies  $\mathbf{X}$  into a multisets  $\langle \mathbf{X} \rangle$  representing the tally as follows:  $\langle \mathbf{X} \rangle = \bigcup_{1 \leq j \leq |\mathbf{X}|} \underbrace{\{j, \dots, j\}}_{\mathbf{X}[j] \text{ times}}$ .

<sup>36</sup> We write  $\mu(k_1, k_2, k_3)$  for the smallest value in  $\{\mu(k_1), \mu(k_2), \mu(k_3)\}$  (cf. [79, pp45]).

```

1  $\mathcal{V} \leftarrow \{1, \dots, n_V\}$ ;
2 for  $i \in \mathcal{V}$  do  $(pk_i, sk_i) \leftarrow \text{Register}(SK_{\mathcal{R}}, i, k_1)$ 
3  $\mathcal{V}' \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V})$ ;
4 for  $i \in \mathcal{V} \setminus \mathcal{V}'$  do  $\beta_i \leftarrow \mathcal{A}()$ ;
5  $BB \leftarrow \{\text{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k_2)\}_{i \in \mathcal{V} \setminus \mathcal{V}'}$ ;
6  $(\mathbf{X}, P) \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$ ;
7  $BB \leftarrow BB \cup \mathcal{A}(BB, \{(pk_i, sk_i)\}_{i \in \mathcal{V} \cap \mathcal{V}'})$ ;
8  $(\mathbf{X}', P') \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$ ;
9 if  $\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}', P') = 1$ 
   $\wedge (\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'} \not\subseteq \langle \mathbf{X}' \rangle \vee |\langle \mathbf{X}' \rangle| - |\langle \mathbf{X} \rangle| > |\mathcal{V}'|)$  then
10 | return 1
11 else
12 | return 0

```

The JCJ-correctness definition implicitly assumes that the tally and associated proof are honestly computed using the Tally algorithm. By comparison, the definition of JCJ-verifiability (Definition 32) does not use this assumption, hence, JCJ-verifiability is intended to assert that voters and auditors can check whether votes have been recorded and tallied correctly. Intuitively, the adversary is assumed to control the tallier and voters, and the security definition challenges the adversary to concoct an election (that is, the adversary generates a bulletin board  $BB$ , a tally  $\mathbf{X}$ , and a proof of tallying  $P$ ) such that verification succeeds and tally  $\mathbf{X}$  differs tally  $\mathbf{X}'$  derived from honestly tallying the bulletin board  $BB$ . It follows that there is at most one verifiable tally that can be derived.

**Definition 32** (JCJ-verifiability). *An election scheme  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  satisfies JCJ-verifiability if for all probabilistic polynomial-time adversary  $\mathcal{A}$ , there exists a negligible function  $\mu$ , such that for all positive integers  $n_C$  and  $n_V$ , and security parameters  $k_1$  and  $k_3$ , we have  $\text{Succ}(\text{Exp-JCJ-Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \leq \mu(k_1, k_2, k_3)$ , where  $\text{Exp-JCJ-Ver}$  is defined as follows:*

```

Exp-JCJ-Ver( $\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3$ ) =
1 for  $1 \leq i \leq n_V$  do  $(pk_i, sk_i) \leftarrow \text{Register}(SK_{\mathcal{R}}, i, k_1)$ 
2  $(BB, \mathbf{X}, P) \leftarrow \mathcal{A}(SK_{\mathcal{T}}, \{(pk_i, sk_i)\}_{i=1}^{n_V})$ ;
3  $(\mathbf{X}', P') \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$ ;
4 if  $\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$ 
  then
5 | return 1
6 else
7 | return 0

```

## APPENDIX J

### PROOFS: JUELS ET AL. ADMIT ATTACKS

This appendix contains proofs demonstrating that the definition of election verifiability by Juels et al. [79] admits collusion and biasing attacks (§VI). We have reported these findings to the original authors [24], [73].

#### A. Proof: Proposition 6

Suppose  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  is an election scheme satisfying JCJ-correctness and JCJ-verifiability. Further suppose  $\text{Stuff}(\Pi, \beta, \kappa) = (\text{Register}, \text{Vote}, \text{Tally}_S, \text{Verify}_S)$ , for some integers  $\beta, \kappa \in \mathbb{N}$ . We prove that  $\text{Stuff}(\Pi, \beta, \kappa)$  satisfies JCJ-correctness and JCJ-verifiability.

We show that  $\text{Stuff}(\Pi, \beta, \kappa)$  satisfies JCJ-correctness by contradiction. Suppose  $\text{Succ}(\text{Exp-JCJ-Cor}(\text{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$  is non-negligible for some  $k_1, k_2, k_3, n_C, n_V$ , and  $\mathcal{A}$ . Hence, there exists an execution of the experiment

$$\text{Exp-JCJ-Cor}(\text{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$$

that satisfies

$$\begin{aligned} &\text{Verify}_S(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}', P') = 1 \\ &\wedge (\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'} \not\subseteq \langle \mathbf{X}' \rangle \vee |\langle \mathbf{X}' \rangle| - |\langle \mathbf{X} \rangle| > |\mathcal{V}'|) \end{aligned}$$

with non-negligible probability, where  $\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'}$  is the set of honest votes,  $(\mathbf{X}, P)$  is the tally of honest votes,  $(\mathbf{X}', P')$  is the tally of all votes,  $\mathcal{V}'$  is a set of corrupt voter identities, and  $BB$  is the bulletin board. Further suppose  $BB_0$  is the bulletin board  $BB$  before adding stuffed ballots. By definition of  $\text{Tally}_S$ , there exist computations

$$(\mathbf{Y}, Q) \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB_0, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

and

$$(\mathbf{Y}', Q') \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

such that  $\mathbf{X} = \text{Add}(\mathbf{Y}, \beta, \kappa)$ ,  $\mathbf{X}' = \text{Add}(\mathbf{Y}', \beta, \kappa)$ , and  $P' = Q'$ . Since  $\kappa \in \mathbb{N}$ , we have  $\langle \mathbf{Y}' \rangle \subseteq \langle \mathbf{X}' \rangle$ . Moreover,  $|\langle \mathbf{X} \rangle| = |\langle \mathbf{Y} \rangle| + \kappa$  and  $|\langle \mathbf{X}' \rangle| = |\langle \mathbf{Y}' \rangle| + \kappa$ , hence,

$$|\langle \mathbf{Y}' \rangle| - |\langle \mathbf{Y} \rangle| = |\langle \mathbf{X}' \rangle| - |\langle \mathbf{X} \rangle|.$$

By definition of  $\text{Verify}_S$  and since  $\mathbf{Y}' = \text{Sub}(\mathbf{X}', \beta, \kappa)$ , there exists a computation

$$v \leftarrow \text{Verify}_0(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{Y}', Q')$$

such that  $v = 1$ . It follows that

$$\begin{aligned} &\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{Y}', Q') = 1 \\ &\wedge (\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'} \not\subseteq \langle \mathbf{Y}' \rangle \vee |\langle \mathbf{Y}' \rangle| - |\langle \mathbf{Y} \rangle| > |\mathcal{V}'|) \end{aligned}$$

with non-negligible probability and, furthermore, we have  $\text{Succ}(\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$  is non-negligible, thereby deriving a contradiction.

We show that  $\text{Stuff}(\Pi, \beta, \kappa)$  satisfies JCJ-verifiability by contradiction. Suppose  $\text{Succ}(\text{Exp-JCJ-Ver}(\text{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$  is non-negligible for some  $k_1, k_3, n_C, n_V$ , and  $\mathcal{A}$ . Hence, there exists an execution of the experiment  $\text{Exp-JCJ-Ver}(\text{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  which satisfies

$$\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$$

with non-negligible probability, where  $(BB, \mathbf{X}, P)$  is an election concocted by the adversary and  $(\mathbf{X}', P')$  is produced by

tallying  $BB$ . By definition of  $\text{Tally}_S$ , there exists a computation

$$(\mathbf{Y}', Q') \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

such that  $\mathbf{X}' = \text{Add}(\mathbf{Y}', \beta, \kappa)$  and  $P' = Q'$ . By definition of  $\text{Verify}_S$ , there exists a computation

$$v \leftarrow \text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \text{Sub}(\mathbf{X}, \beta, \kappa), P)$$

such that  $v = 1$ . Let the adversary  $\mathcal{B}$  be defined as follows: given input  $K$  and  $S$ , the adversary  $\mathcal{B}$  computes

$$(BB, \mathbf{X}, P) \leftarrow \mathcal{A}(K, S)$$

and outputs  $(BB, \text{Sub}(\mathbf{X}, \beta, \kappa), P)$ . We have an execution of the experiment  $\text{Exp-JCJ-Ver}(\text{Stuff}(\Pi, \beta, \kappa), \mathcal{B}, n_C, n_V, k_1, k_2, k_3)$  that concocts the election  $(BB, \text{Sub}(\mathbf{X}, \beta, \kappa), P)$  and tallying  $BB$  produces  $(\mathbf{Y}', Q')$  such that

$$\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \text{Sub}(\mathbf{X}, \beta, \kappa), P) = 1$$

with non-negligible probability. Moreover, since  $\mathbf{X} \neq \mathbf{X}'$  and  $\mathbf{Y}' = \text{Sub}(\mathbf{X}', \beta, \kappa)$ , we have  $\text{Sub}(\mathbf{X}, \beta, \kappa) \neq \mathbf{Y}'$  with non-negligible probability. It follows immediately that  $\text{Succ}(\text{Exp-JCJ-Cor}(\Pi, \mathcal{B}, n_C, n_V, k_1, k_2, k_3))$  is non-negligible, thus deriving a contradiction and concluding our proof.  $\square$

### B. Proof: Proposition 7

We define key leakage before proving Proposition 7.

**Definition 33** (Key leakage). *An election scheme  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  does not leak the tallier's private key if for all positive integers  $n_C$  and  $n_V$ , security parameters  $k_1$  and  $k_3$ , and probabilistic polynomial-time adversary  $\mathcal{A}$ , we have  $\text{Succ}(\text{Exp-leak}(\Pi, \mathcal{A}, k_1, k_3, n_C, n_V))$  is negligible, where  $\text{Exp-leak}(\cdot)$  is defined as follows:*

$$\text{Exp-leak}(\Pi, \mathcal{A}, k_1, k_3, n_C, n_V) =$$

```

1 for  $1 \leq i \leq n_V$  do  $(pk_i, sk_i) \leftarrow \text{Register}(SK_{\mathcal{R}}, i, k_1)$ ;
2  $BB \leftarrow \mathcal{A}(sk_1, \dots, sk_{n_V})$ ;
3  $(\mathbf{X}, P) \leftarrow \text{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$ ;
4 if  $\mathcal{A}(\mathbf{X}, P) = SK_{\mathcal{T}}$  then
5 | return 1
6 else
7 | return 0
```

*Proof of Proposition 7.* Suppose  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  is an election scheme satisfying JCJ-correctness and JCJ-verifiability that does not leak the tallier's private key, and  $\text{Backdoor}(\Pi) = (\text{Register}, \text{Vote}, \text{Tally}_B, \text{Verify}_B)$ . Since  $\Pi$  does not leak the tallier's private key, the experiment  $\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  produces a bulletin board  $BB$  such that  $SK_{\mathcal{T}} \in BB$  with negligible probability, for all  $k_1, k_2, k_3, n_C, n_V$  and  $\mathcal{A}$ . It follows that an execution of the experiment  $\text{Exp-JCJ-Cor}(\text{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  is an execution of the experiment  $\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  with non-negligible probability and, hence,  $\text{Backdoor}(\Pi)$  satisfies JCJ-correctness.

We show that  $\text{Backdoor}(\Pi)$  satisfies JCJ-verifiability by contradiction. Suppose  $\text{Succ}(\text{Exp-JCJ-Ver}(\text{Backdoor}(\Pi), \mathcal{A},$

$n_C, n_V, k_1, k_2, k_3))$  is non-negligible for some  $k_1, k_3, n_C, n_V$ , and  $\mathcal{A}$ . Hence, there exists an execution of the experiment  $\text{Exp-JCJ-Ver}(\text{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  which satisfies

$$\text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$$

with non-negligible probability, where  $(BB, \mathbf{X}, P)$  is an election concocted by the adversary and  $(\mathbf{X}', P')$  is produced by tallying  $BB$ . If there is one and only one  $\mathbf{Y}$  such that  $(SK_{\mathcal{T}}, \mathbf{Y}) \in BB$ , then  $\mathbf{X}' = \mathbf{Y}$  by definition of  $\text{Tally}$  and  $\mathbf{X} = \mathbf{Y}$  by definition of  $\text{Verify}$ , otherwise, the execution of the experiment  $\text{Exp-JCJ-Cor}(\text{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  is an execution of the experiment  $\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$  and, hence,

$$\begin{aligned} & \text{Succ}(\text{Exp-JCJ-Ver}(\text{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ &= \text{Succ}(\text{Exp-JCJ-Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)). \end{aligned}$$

In both cases we derive a contradiction, thereby concluding our proof.  $\square$

### C. Proof sketch: Proposition 8

Suppose  $\Pi = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify})$  is an election scheme satisfying JCJ-correctness and JCJ-verifiability. Further suppose  $\text{Bias}(\Pi, Z) = (\text{Register}, \text{Vote}, \text{Tally}, \text{Verify}_R)$ , for some set of vectors  $Z$ . By definition of  $\text{Verify}_R$ , we have

$$\text{Verify}_R(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1$$

implies the existence of a computation

$$v \leftarrow \text{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P)$$

such that  $v = 1$  with non-negligible probability, for all  $PK_{\mathcal{T}}, BB, n_C, \mathbf{X}$ , and  $P$ . It follows that

$$\begin{aligned} & \text{Succ}(\text{Exp-JCJ-Cor}(\text{Bias}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ & \leq \text{Succ}(\text{Exp-JCJ-Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \end{aligned}$$

and

$$\begin{aligned} & \text{Succ}(\text{Exp-JCJ-Ver}(\text{Bias}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ & \leq \text{Succ}(\text{Exp-JCJ-Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \end{aligned}$$

for all  $k_1, k_2, k_3, n_C, n_V$ , and  $\mathcal{A}$ . Hence,  $\text{Bias}(\Pi, Z)$  satisfies JCJ-correctness and JCJ-verifiability.  $\square$

## 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 Hrițcu, 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, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO'98: 18th International Cryptology Conference*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998.
- [9] 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.
- [10] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 519–536. Springer, 1999.
- [11] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. Cryptology ePrint Archive, Report 2006/228, 2006.
- [12] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1996.
- [13] Josh Benaloh. Simple Verifiable Elections. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.
- [14] Josh Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.
- [15] 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.
- [16] 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](http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html) (accessed 7 May 2014), Sept 2010.
- [17] 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.
- [18] David Bernhard. *Zero-Knowledge Proofs in Theory and Practice*. PhD thesis, Department of Computer Science, University of Bristol, 2014.
- [19] 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.
- [20] 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.
- [21] David Bernhard, Olivier Pereira, and Bogdan Warinschi. On Necessary and Sufficient Conditions for Private Ballot Submission. Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b), 2012.
- [22] 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.
- [23] 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.
- [24] Dario Catalano. Private communication, Paris, France, 10th October 2013.
- [25] 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.
- [26] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
- [27] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.
- [28] 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.
- [29] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 200–212. Springer, 1987.
- [30] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In *CRYPTO'92: 12th International Cryptology Conference*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.
- [36] 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.
- [37] Véronique Cortier and David Galindo. Private communication, Nancy, France, 13th June 2013.
- [38] Véronique Cortier, David Galindo, Stéphane 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.
- [39] 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.
- [40] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.
- [41] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO'94: 14th International Cryptology Conference*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
- [42] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Authority Secret-Ballot Elections with Linear Work. In *EUROCRYPT'96: 15th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.
- [43] 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.
- [44] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In

- CRYPTO'98: 18th International Cryptology Conference*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.
- [45] Chris Culnane and Steve A. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *CSF'14: 27th Computer Security Foundations Symposium*, pages 169–183. IEEE Computer Society, 2014.
- [46] Ivan Damgård. On  $\Sigma$ -protocols, 2010. Available from <http://www.daimi.au.dk/~ivan/Sigma.pdf>.
- [47] 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.
- [48] 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.
- [49] Jannik Dreier, Rosario Giustolisi, Ali Kassem, Pascal Lafourcade, and Gabriele Lenzini. On the Verifiability of (Electronic) Exams. Technical Report TR-2014-2, Verimag, 2014.
- [50] 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.
- [51] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
- [52] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
- [53] 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.
- [54] 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.
- [55] David Galindo and Véronique Cortier. Private email communication, 19th June 2013.
- [56] David Galindo and Véronique Cortier. Private email communication, Summer/Autumn 2014.
- [57] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 295–310, 1999.
- [58] 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).
- [59] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, Cambridge, UK, 2001.
- [60] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In *STOC'82: 14th Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM Press, 1982.
- [61] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
- [62] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989.
- [63] 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.
- [64] Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT'02: 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.
- [65] 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.
- [66] Carmit Hazay and Yehuda Lindell. Sigma protocols and efficient zero-knowledge. In *Efficient Secure Two-Party Protocols*, Information Security and Cryptography, chapter 6, pages 147–175. Springer Berlin Heidelberg, 2010.
- [67] James Heather and David Lundin. The Append-Only Web Bulletin Board. In *FAST'08: 5th International Workshop on Formal Aspects in Security and Trust*, volume 5491 of *LNCS*, pages 242–256. Springer, 2008.
- [68] 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.
- [69] 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.
- [70] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. In *WOTE'06: Workshop on Trustworthy Elections*, 2006.
- [71] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. *Mathematical and Computer Modelling*, 48(9-10):1628–1645, 2008.
- [72] IACR Elections. <http://www.iacr.org/elections/> (accessed 7 May 2014), 2013.
- [73] Markus Jakobsson. Private communication, New Orleans, USA, 27th June 2013.
- [74] 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.
- [75] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *11th USENIX Security Symposium*, pages 339–353, 2002.
- [76] 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.
- [77] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. *Cryptology ePrint Archive*, Report 2002/165, 2002.
- [78] 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.columbia.rsa.com/rsalabs/staff/ajuels/publications/Coercion/Coercion.pdf> (accessed 7 May 2014).
- [79] 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.
- [80] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.
- [81] 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.
- [82] Aggelos Kiayias. Electronic voting. In *Handbook of Financial Cryptography and Security*, chapter 3. Chapman and Hall/CRC, 2010.
- [83] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 468–498. Springer Berlin Heidelberg, 2015.
- [84] 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.
- [85] 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.
- [86] Ralf Kuesters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. *Cryptology ePrint Archive*, Report 2010/236 (version 20150202:163211), 2015.
- [87] Ralf Küsters. Private email communication, 24th June 2014.
- [88] Ralf Küsters. Private email communication, October/November 2014.

- [89] 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.
- [90] 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.
- [91] 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>.
- [92] 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.
- [93] 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.
- [94] Philip MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385–400. Springer Berlin Heidelberg, 2002.
- [95] 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.
- [96] 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.
- [97] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Unpublished manuscript, 2004.
- [98] NIST. Secure Hash Standard (SHS). FIPS PUB 180-4, Information Technology Laboratory, National Institute of Standards and Technology, March 2012.
- [99] Helios Princeton Elections. <https://princeton.heliosvoting.org/> (accessed 7 May 2014), 2012.
- [100] Participants of the Dagstuhl Conference on Frontiers of E-Voting. *Dagstuhl Accord*, 2007. <http://www.dagstuhlaccord.org/> (accessed 7 May 2014).
- [101] R. A. Peters. A secure bulletin board. Master's thesis, Technische Universiteit Eindhoven, June 2005.
- [102] Kazuo 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.
- [103] Daniel Sandler and Dan S. Wallach. Casting votes in the Auditorium. In *EVT'07: Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2007. USENIX Association.
- [104] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89: 9th International Cryptology Conference*, volume 435 of *LNCS*, pages 239–252. Springer, 1990.
- [105] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Trans. Comput. Logic*, 6(3):634–671, July 2005.
- [106] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.
- [107] 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.
- [108] Ben Smyth and David Bernhard. Ballot secrecy with malicious bulletin boards. Technical Report 2014/822, Cryptology ePrint Archive, 2014.
- [109] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjeh. 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.
- [110] *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](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).
- [111] 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.