Election Verifiability for Helios under Weaker Trust Assumptions*

Véronique Cortier[1], David Galindo[1], Stéphane Glondu[2], and Malika Izabachène[1,3]

[1] LORIA - CNRS, France
[2] INRIA Nancy Grand Est, France
[3] École Polytechnique Féminine, France

Abstract. Most electronic voting schemes aim at providing verifiability: voters should trust the result without having to rely on some authorities. Actually, even a prominent voting system like Helios cannot fully achieve verifiability since a dishonest bulletin board may add ballots. This problem is called *ballot stuffing*.

In this paper we give a definition of verifiability in the computational model to account for a malicious bulletin board that may add ballots. Next, we provide a generic construction that transforms a voting scheme that is verifiable against an honest bulletin board and an honest registration authority (*weak verifiability*) into a verifiable voting scheme under the weaker trust assumption that the registration authority and the bulletin board are *not simultaneously* dishonest (*strong verifiability*). This construction simply adds a registration authority that sends private credentials to the voters, and publishes the corresponding public credentials.

We further provide simple and natural criteria that imply weak verifiability. As an application of these criteria, we formally prove the latest variant of Helios by Bernhard, Pereira and Warinschi weakly verifiable. By applying our generic construction we obtain a Helios-like scheme that has ballot privacy and strong verifiability (and thus prevents ballot stuffing). The resulting voting scheme, Helios-C, retains the simplicity of Helios and has been implemented and tested.

Keywords: voting protocols, individual verifiability, universal verifiability, ballot stuffing, ballot privacy, Helios.

1 Introduction

Ideally, a voting system should be both private and verifiable. Privacy ensures that no one knows that a certain voter has voted in a particular way. Verifiability ensures that voters should be able to check that, even in the presence of dishonest tallying authorities, their ballots contribute to the outcome (individual verifiability) and that the the published result corresponds to the intended votes of the voters (universal verifiability). One leading voting system designed to achieve both privacy and verifiability is Helios [1], based on a classical voting system proposed by Cramer, Gennaro and Schoenmakers [2] with variants proposed by Benaloh [3]. Helios is an open-source voting system that has

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865.

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part II, LNCS 8713, pp. 327–344, 2014.

been used several times to run real-world elections, including the election of the president of the University of Louvain-La-Neuve and the election of the 2010, 2011, and 2012 new board directors of the International Association for Cryptographic Research (IACR) [4]. Helios has been shown to ensure ballot privacy for successively stronger notions of privacy and more accurate implementations [5–7].

The remaining question is whether the result of an election run through Helios does correspond to the votes cast by the voters. Put in other words, is Helios verifiable? According to Juels, Catalano and Jakobsson (JCJ) definition [8], Helios is individually and universally verifiable[1], although we are not aware of any proof of verifiability in a computational model. In fact, Bernhard, Pereira and Warinschi (BPW) [7] showed recently that existing Helios versions [9] are not verifiable due to the use of a weak version of the Fiat-Shamir transformation in the non-interactive zero-knowledge proofs of ballot well-formedness. They showed that when the standard version of Fiat-Shamir is used, then Helios has ballot privacy but they do not prove verifiability. The forthcoming Helios version 4.0 is planned to incorporate these changes [9].

Still, JCJ's definition assumes the bulletin board to be honest: an attacker may cast dishonest ballots on the behalf of dishonest voters but no extra ballots may be added nor deleted. This means for example that the result of the election of the 2012 board of the IACR can be trusted only under the assumption that the election server was neither dishonest nor attacked, during the whole duration of the election. This is a rather unsatisfactory assumption, since adding a few extra ballots may easily change the outcome of an election. In the case of Helios, this is mitigated by the fact that voters' identities are public. If the bulletin board adds ballots, it has to tell which voters are supposed to have cast these ballots. Thus hopefully, these voters should notice that the server wrongly cast ballots on their names and would complain. Such complaints are however not guaranteed since absentees typically do not care much about the election. Things may be even worse. In some countries (like France), whether someone voted or not is a private information (that can be accessed only by voters of the same precinct, through a rather heavy procedure). It is therefore forbidden to publicly reveal the identities of the voters who cast a vote. Moreover, publishing voters identities compromises privacy in the future: once the public key of the election will be broken (say in 20 years), everyone will learn the vote of each voter. A simple alternative consists in removing the disclosure of voters' identities. This variant of Helios remains perfectly practical and of course still preserves ballot privacy. But it then becomes completely straightforward for a corrupted bulletin board to add as many ballots as needed to change the legitimate election result.

Election Verifiability under Weaker Trust Assumptions. We first provide an extension of the definition of individual and universal verifiability by Juels, Catalano and Jakobsson [8], that accounts for ballot stuffing. Throughout the paper we will sometimes use *verifiability* to refer to *individual and universal verifiability*. Intuitively, a voting scheme is *verifiable* if the result corresponds to the votes of

[1] JCJ uses the terms *correctness* and *verifiability*, which we rename as *individual and universal verifiability* and *tally uniqueness* respectively, as we think the latter terminology matches better the e-voting literature and it is also more accurate.

- all honest voters that have checked that their vote was cast correctly (in Helios, this amounts into checking that the encrypted vote appears on the bulletin board);
- at most n valid votes where n is the number of corrupted voters (i.e. the attacker may only use the corrupted voters to cast valid votes);
- a subset of the votes cast by honest voters that did not check their vote was cast correctly (in practice, many voters do not perform any check).

As in [8], this definition requires the tally function to admit *partial tallying* (that is, it is possible to compute the tally by blocks and then retrieve the final result). This is satisfied by most election systems, notably those consisting on counting the number of votes that every candidate from a given list received, and those whose outcome is the multiset of cast votes.

Our first main contribution is a generic construction that transforms any verifiable voting scheme that assumes both the registration authority and the bulletin board honest, into a verifiable voting scheme under the weaker trust assumption that the registration authority and the bulletin board are not *simultaneously* dishonest. We show that our transformation also turns ballot privacy and tally uniqueness (as defined in Section 3.3) w.r.t. honest bulletin board and registration authority, into ballot privacy and tally uniqueness w.r.t. non simultaneously dishonest bulletin board and registration authority. Throughout the paper we will sometimes use *strong verifiability* to refer to *individual and universal verifiability against non simultaneously dishonest bulletin board and registration authority*.

We stress that verifiability cannot come without trust assumptions: the key issue relies on the fact that some mechanism is necessary to *authenticate* voters, that is, to make sure that Bob is not voting in the name of Alice. In Helios-like protocols, the bulletin board is the only authority that controls the right to vote. It may therefore easily stuff itself, that is, it may easily add ballots. To control the bulletin board, it is necessary to consider an additional authority. In our solution, a so-called *registrar* authority, provides each voter with a private credential (actually a signing key) that has a public part (the verification key). The set of all public credentials is public and, in particular, known to the bulletin board. Then each voter simply signs his ballot with his private credential. Note that the association between a public credential and the corresponding voter's identity does not need to be known and actually, should not be disclosed to satisfy e.g. the French requirements regarding voting systems. It is also possible to have the registration authority to generate the credentials off-line and to distribute them using a non-digital channel, e.g. snail mail. This minimizes the risk of Internet-based attacks against the registration authority. We have designed our solution having in mind the guidelines set for the e-voting setup used for the expatriates at the 2012 French legislative elections [10].

The advantage of our approach relies on its simplicity: the additional authority is only responsible for generating and distributing the credentials of the voters. Once it is done, it can erase these records. It consists on one offline layer added on top of the existing voting protocol; therefore it needs not to be changed and its infrastructure is kept. In particular, our solution does not require any additional server.

We have also considered the possibility of using anonymous credentials [11]. Our preliminary conclusion discards a direct application in our transformation. This is due

to the fact that anonymous credentials allow its owners to unlinkably "show" the same credential multiple times. In our case this property potentially allows a voter to vote several times without being detected, and then verifiability cannot be achieved.

Criteria for Universal Verifiability. Since proving verifiability against cheating tallying authorities, even assuming honest bulletin board and registration authority, may not be easy, we provide a simple and natural criteria that implies verifiability. We show that any *correct* and *accurate* voting protocol with *tally uniqueness* is universally verifiable (w.r.t. an honest bulletin board). Correctness accounts for the natural property that the tally of just honestly cast ballots should always yield the expected result (typically the sum of the votes). Accuracy ensures that any ballot (possibly dishonest) that passes the verification check (e.g. valid proof, well-formedness of the ballots) corresponds to a valid vote. Tally uniqueness ensures that two different results cannot be announced for a single election. Our criteria are satisfied in particular by Helios and we expect it to be satisfied by many existing voting protocols. As a result we provide the *first proof* of verifiability for the Helios-BPW voting scheme [7] in a computational model.

A Verifiable Helios-Like Scheme That Prevents Ballot Stuffing. By applying our generic construction to Helios-BPW we obtain a voting scheme, that we name as Helios with Credentials (Helios-C), which is verifiable against cheating tallying authorities under the weak assumption that the bulletin board and the registration authority are not simultaneously dishonest. Helios-C is ballot private if the tallying authority behaves honestly. We have implemented Helios-C and used it in a mock election.

Related Work. To the best of our knowledge, the only proofs of verifiability for Helios have been conducted in abstract models. Delaune, Kremer and Ryan [12] define individual and universal verifiability in a symbolic model and prove that Helios satisfy both. Like for all symbolic models, the cryptographic primitives are abstracted by terms and are not analyzed. Küsters *et al.* have put forward quantitative measurements of verifiability and accountability in [13–15] that take into account ballot stuffing. In particular, [15] gives accountability measures on several abstractions of Helios. In contrast to [15], our verifiability framework is less expressive, but on the contrary we prove verifiability in the computational model. Verifiability proofs like those of [12] and [13–15] can typically not detect flaws that on the cryptographic primitives, like those found by Bernhard, Pereira and Warinschi [7]. Groth [16] studies a generalized version of Helios in the Universal Composability framework, but it does not address universal verifiability.

2 Syntax of a Voting System

Election systems typically involve several entities. For the sake of simplicity we consider each entity to consist of only one individual but all of them could be thresholdized.

1. *Election Administrator*: Denoted by $\mathcal{E}$, is responsible for setting up the election. It publishes the identities id of eligible voters, the list of candidates and the result function ρ of the election (typically counting the number of votes every candidate received).
2. *Registrar*: Denoted by $\mathcal{R}$, is responsible for distributing secret credentials to voters and registering the corresponding public credentials.

3. *Trustee:* Denoted by $\mathcal{T}$, is in charge of tallying and publishing a final result.
4. *Voters*: The eligible voters $id_1, \ldots, id_\tau$ are participating in the election.
5. *Bulletin board manager*: Denoted by $\mathcal{B}$, is responsible for processing ballots and storing valid ballots in the bulletin board BB.

2.1 Voting Algorithms

We continue by describing the syntax for an electronic voting protocol that we will be using thorough the paper. The syntax below considers *single-pass* schemes, namely systems where voters only have to post a single message in the board. A voting protocol is always relative to a family of result functions $\mathcal{R} = \{\rho_\tau\}_{\tau \geq 1}$ for $\tau \in \mathbb{N}$, where $\rho_\tau : \mathbb{V}^\tau \rightarrow \mathbf{R}$, $\mathbf{R}$ is the result space and $\mathbb{V}$ is the set of admissible votes. A voting protocol $\mathcal{V} = (\mathsf{Setup}, \mathsf{Credential}, \mathsf{Vote}, \mathsf{Validate}, \mathsf{Box}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{Verify})$ consists of eight algorithms whose syntax is as follows:

$\mathsf{Setup}(1^\lambda)$ on input a security parameter 1^λ, outputs an election public/secret pair $(\mathbf{pk}, \mathbf{sk})$, where $\mathbf{pk}$ typically contains the public key of the election and/or a list of credentials L. We assume $\mathbf{pk}$ to be an implicit input of the remaining algorithms.

$\mathsf{Credential}(1^\lambda, id)$ on inputs a security parameter 1^λ and an identifier id, outputs the secret part of the credential usk_{id} and its public credential upk_{id}, where upk_{id} is added to the list $L = \{\mathsf{upk}_{id}\}$.

$\mathsf{Vote}(id, \mathsf{upk}, \mathsf{usk}, v)$ is used by voter id to cast his choice $v \in \mathbb{V}$. It outputs a ballot b, which may/may not include the identifier id or the public credential upk. The ballot b is sent to the bulletin board through an authenticated channel. At some point, the voter may reach a state where he/she considers his/her vote has been counted, typically after having run the algorithm $\mathsf{VerifyVote}$ defined below. The voter then set $\mathsf{CheckedVoter}(id, v, b)$ to true.

$\mathsf{Validate}(b)$ on input a ballot b returns 1 for well-formed ballots and 0 otherwise.

$\mathsf{Box}(\mathsf{BB}, b)$ takes as inputs the bulletin board BB and a ballot b and outputs an updated BB. Typically, this algorithm performs some checks on b with respect to the contents of BB and, possibly, a local state st. Depending on these checks, BB and st are updated; in any case BB remains unchanged if $\mathsf{Validate}(b)$ rejects (that is returns 0). We say that BB is well-formed if $\mathsf{Validate}(b) = 1$ for every $b \in \mathsf{BB}$.

$\mathsf{VerifyVote}(\mathsf{BB}, id, \mathsf{upk}, \mathsf{usk}, b)$ is a typically light algorithm intended to the voters, for checking that their ballots will be included in the tally. On inputs the board BB, a ballot b, and the voter's identity and credentials id, usk, upk, returns 1 or 0.

$\mathsf{Tally}(\mathsf{BB}, \mathbf{sk})$ takes as input the bulletin board BB and the secret key $\mathbf{sk}$. After some checks, it outputs the tally ρ, together with a proof of correct tabulation Π. Possibly, $\rho = \perp$, meaning the election has been declared invalid.

$\mathsf{Verify}(\mathsf{BB}, \rho, \Pi)$ on inputs the bulletin board BB, and a pair (ρ, Π), checks whether Π is a valid proof of correct tallying for ρ. It returns 1 if so; otherwise it returns 0.

The exact implementation of the algorithms of course depends on the voting protocol under consideration. In Helios, the authenticated channel is instantiated by a login and a password and we have $\mathsf{upk}_{id} \in \{\emptyset, id, pid\}$ depending on the variants. $\mathsf{upk}_{id} = id$ corresponds to the standard case where the identity of the voter is appended to the

ballot and displayed on the bulletin board. $\mathsf{upk}_{id} = pid$, where pid is a pseudonym on identity id, corresponds to the case where only pseudonyms are displayed, to provide more privacy to the voters. Finally, $\mathsf{upk}_{id} = \emptyset$ corresponds to the case where only the raw ballot is displayed on the bulletin board. We provide in Section 5 a complete description of the Helios protocol and our variant of it.

2.2 Correctness

Next we define the minimal requirement, called *correctness*, that any voting protocol must satisfy. It simply requires that honest executions yield the expected outcome, that is, honestly cast ballots are accepted to the BB (and pass the verification checks) and that, in an honest setting, the tally procedure always yields the expected outcome (that is, the result function). Let $\mathsf{BB} := \{\emptyset\}$. A voting scheme is *correct* if: (1) For $i \in \{1, \ldots, \tau\}$, it holds that $\mathsf{Validate}(b_i) = 1$, $\mathsf{VerifyVote}\big(\mathsf{Box}(\mathsf{BB}, b_i), id_i, \mathsf{upk}_i, \mathsf{usk}_i, b_i\big) = 1$, and $\mathsf{Box}(\mathsf{BB}, b_i) = \mathsf{BB} \cup \{b_i\}$, where $b_i \leftarrow \mathsf{Vote}(id_i, \mathsf{upk}_i, \mathsf{usk}_i, v_i)$ for some $v_i \in \mathbb{V}$; (2) $\mathsf{Tally}(\{b_1, \ldots, b_\tau\}, \mathbf{sk})$ outputs $(\rho(v_1, \ldots, v_\tau), \Pi)$; and (3) $\mathsf{Verify}(\{b_1, \ldots, b_\tau\}, \rho(v_1, \ldots, v_\tau), \Pi) = 1$. The above properties can be relaxed to hold only with overwhelming probability.

3 Verifiability Definitions

In this section we give individual and universal verifiability definitions in which the election administrator is honest, but trustee and voters are assumed to be dishonest. As emphasized in Introduction, verifiability partly relies on the authentication of the voters. There are various ways to authenticate voters, but in each case, it requires some trust assumptions. Our minimal trust assumption is that the registrar and the bulletin board are *not simultaneously* dishonest. We further define a property, that we call *tally uniqueness*, where no party is assumed be honest (except for the election administrator).

Partial Tallying. We focus on voting protocols that admit *partial tallying*. This property is specified by two natural requirements usually satisfied in most election scenarios. Firstly, the result function $\rho : \mathbb{V}^\tau \to \mathbf{R}$ for $\mathcal{V}$ must admit *partial counting*, namely $\rho(S_1 \cup S_2) = \rho(S_1) \star_{\mathbf{R}} \rho(S_2)$ for any two lists S_1, S_2 containing sequences of elements $v \in \mathbb{V}$ and where $\star_{\mathbf{R}} : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ is a commutative operation. For example, the standard result function that counts the number of votes per candidate admits partial counting. Secondly, the algorithm Tally must admit *partial tallying*, *i.e.* let $(\rho_1, \Pi_1) \leftarrow \mathsf{Tally}(\mathsf{BB}_1, \mathbf{sk})$ and $(\rho_2, \Pi_2) \leftarrow \mathsf{Tally}(\mathsf{BB}_2, \mathbf{sk})$. Let $(\rho, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_1 \cup \mathsf{BB}_2, \mathbf{sk})$ with ρ different from invalid and BB_1 and BB_2 disjoint. Then, $\rho = \rho_1 \star_{\mathbf{R}} \rho_2$, with overwhelming probability.

3.1 Strong Verifiability

We say that a voting scheme achieves *strong verifiability* if it has individual and universal verifiability under the sole trust assumption that the registrar and the bulletin board are *not simultaneously* dishonest. More formally, a voting scheme has strong verifiability if it has *verifiability against a dishonest bulletin board* and *verifiability against a dishonest registrar*. These are defined below.

Election Verifiability against a Dishonest Bulletin Board. This is an extension of security property already addressed in [8, 17]. Our novelty is that we assume the bulletin board to be possibly dishonest, and in particular it may stuff ballots in the name of voters who did never cast a vote. Of course, a verifiable protocol should forbid or at least detect such a malicious behavior.

We consider an adversary against individual and universal verifiability that is allowed to corrupt trustee, users and bulletin board. Only the registration authority is *honest*. More precisely, for the bulletin board, we let the adversary replace or delete any ballot. The adversary only looses control on the bulletin board once the voting phase ends and before the tallying starts. Indeed, at this point it is assumed that everyone has the same view of the public BB.

Let L denote the set of public credentials, $\mathcal{U}$ the set of public/secret credentials pairs, and $\mathcal{CU}$ the set of corrupted users. The adversary can query oracles $\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corrupt}$ and $\mathcal{O}\mathsf{vote}$. Let HVote contain triples (id, v, b) that have been output by $\mathcal{O}\mathsf{vote}$ (if voter id voted multiple times, only the last ballot is retained); while the list $\mathsf{Checked}$ consists of all pairs $(id, v, b) \in \mathsf{HVote}$ such that $\mathsf{CheckedVoter}(id, v, b) = 1$, that is, $\mathsf{Checked}$ corresponds to voters that have checked that their ballots will be counted (typically running $\mathsf{VerifyVote}$).

- $\mathcal{O}\mathsf{reg}(id)$: invokes algorithm $\mathsf{Credential}(\lambda, id)$, it returns upk_{id} and keeps usk_{id} secret. It also updates the lists $L = L \cup \{\mathsf{upk}_{id}\}$ and $\mathcal{U} = \mathcal{U} \cup \{(id, \mathsf{upk}_{id}, \mathsf{usk}_{id})\}$.
- $\mathcal{O}\mathsf{corrupt}(id)$: firstly, checks if an entry $(id, *, *)$ appears in $\mathcal{U}$; if not, stops. Else, outputs $(\mathsf{upk}_{id}, \mathsf{usk}_{id})$ and updates $\mathcal{CU} = \mathcal{CU} \cup \{(id, \mathsf{upk}_{id})\}$.
- $\mathcal{O}\mathsf{vote}(id, v)$: if $(id, *, *) \notin \mathcal{U}$ or $(id, *) \in \mathcal{CU}$ or $v \notin \mathbb{V}$, aborts; else returns $b = \mathsf{Vote}(id, \mathsf{upk}_{id}, \mathsf{usk}_{id}, v)$ and replaces any previous entry $(id, *, *)$ in HVote with (id, v, b).

Any voting scheme should guarantee that the result output by $\mathsf{Tally}(\mathsf{BB}, \mathbf{sk})$ counts the actual votes cast by honest voters. In particular an adversary controlling a subset of eligible voters, the trustee and the bulletin board, should not be able to alter the output of

Experiment $\mathsf{Exp}^{\mathsf{verb}}_{\mathcal{A},\mathcal{V}}(\lambda)$

(1) $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathsf{Setup}(\lambda)$

(2) $(\mathsf{BB}, \rho, \Pi) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{vote}}$

(3) if $\mathsf{Verify}(\mathsf{BB}, \rho, \Pi) = 0$ return 0

(4) if $\rho = \perp$ return 0

(5) if $\exists\ (id_1^A, v_1^A, *), \ldots, (id_{n_A}^A, v_{n_A}^A, *) \in \mathsf{HVote} \backslash \mathsf{Checked}$

$\exists\ v_1^B, \ldots, v_{n_B}^B \in \mathbb{V}$ s.t. $0 \leq n_B \leq |\mathcal{CU}|$

s.t. $\rho = \rho\left(\{v_i^E\}_{i=1}^{n_E}\right) \star_{\mathbf{R}} \rho\left(\{v_i^A\}_{i=1}^{n_A}\right) \star_{\mathbf{R}} \rho\left(\{v_i^B\}_{i=1}^{n_B}\right)$

return 0 else return 1

where $\mathsf{Checked} = \{(id_1^E, v_1^E, b_1^E), \ldots, (id_{n_E}^E, v_{n_E}^E, b_{n_E}^E)\}$

Fig. 1. Verifiability against a malicious bulletin board

the tally so that honest votes are not counted in ρ. More precisely, verifiability against a dishonest board shall guarantee that ρ as output by the algorithm Tally actually counts:

1. votes cast by honest voters who *checked* that their ballot appeared in the bulletin board (corresponds to $\{v_i^E\}_{i=1}^{n_E}$ in Figure 1);
2. a subset of the votes cast by honest voters who *did not check* this. Indeed it can not be ensured that ρ counted their votes but it might still be the case that some of their ballots were not deleted by the adversary (corresponds to $\{v_i^A\}_{i=1}^{n_A}$ in Figure 1).
3. For corrupted voters, it is only guaranteed that the adversary cannot cast more ballots than users were corrupted, and that ballots produced by corrupted voters contribute to ρ only with admissible votes $v \in \mathbb{V}$ (corresponds to $\{v_i^B\}_{i=1}^{n_B}$).

The verifiability against a malicious board game is formally given by experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verb}}$ in Figure 1. We say that a voting protocol $\mathcal{V}$ is *verifiable against a dishonest board* if there exists a negligible function $\nu(\lambda)$ such that, for any PPT adversary $\mathcal{A}$, $\mathsf{Succ}_{\mathcal{V}}^{\mathsf{verb}}(\mathcal{A}) = \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verb}}(\lambda) = 1\right] < \nu(\lambda)$.

Election Verifiability against a Dishonest Registration Authority. The corresponding experiment $\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verg}}$ defining verifiability against a malicious registration authority and malicious trustee and voters, but honest bulletin board, is very similar to the experiment in Figure 1. The adversary has access to oracles $\mathcal{O}\mathsf{vote}(id, v)$ and $\mathcal{O}\mathsf{corrupt}(id)$ as before, and is additionally given access to an oracle $\mathcal{O}\mathsf{cast}(id, b)$, which runs $\mathsf{Box}(\mathsf{BB}, b)$. This models the fact that the adversary cannot delete nor add ballots anymore since the bulletin box is now honest. However, the adversary is not given in this experiment access to the $\mathcal{O}\mathsf{reg}$ oracle, since it controls the registrar and thus can register users arbitrarily, even with malicious credentials. The adversary uses $\mathcal{O}\mathsf{corrupt}(id)$ to define voter id as a corrupted user, i.e. voter id's actions are under the control of the adversary.

In $\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verg}}$, the adversary does not output BB, since the bulletin board is honest. Note that a dishonest registration authority may prevent some voters from voting by providing wrong credentials. Depending on the protocol, voters may not notice it, therefore some honestly cast ballots may be discarded.

We say that $\mathcal{V}$ is *verifiable against a dishonest registration authority* if there exists a negligible function $\nu(\lambda)$ such that, $\mathsf{Succ}_{\mathcal{V}}^{\mathsf{verg}}(\mathcal{A}) = \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verg}}(\lambda) = 1\right] < \nu(\lambda)$, for any PPT adversary $\mathcal{A}$.

3.2 Weak Verifiability

We say that a voting scheme has *weak verifiability* if it has individual and universal verifiability assuming that the bulletin board and the registration authority are *both* honest. That is, an adversary in the weak verifiability game can only corrupt a subset of voters and the trustee.

The experiment $\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verw}}$ defining *weak verifiability*, is a variation of the experiment $\mathsf{Exp}_{\mathcal{A},\mathcal{V}}^{\mathsf{verg}}$. In this case, the adversary can only add ballots to the box via $\mathcal{O}\mathsf{cast}$ (so it

cannot stuff the ballot box nor delete ballots). The adversary is only allowed to register voters through $\mathcal{O}\mathsf{reg}$, and can only access voters' secret credentials by calling the $\mathcal{O}\mathsf{corrupt}$ oracle. We say that a voting protocol $\mathcal{V}$ is *weakly verifiable* if there exists a negligible function $\nu(\lambda)$ such that, $\mathsf{Succ}^{\mathsf{verw}}_{\mathcal{V}}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{verw}}_{\mathcal{A},\mathcal{V}}(\lambda) = 1\right] < \nu(\lambda)$, for any PPT adversary $\mathcal{A}$.

3.3 Tally Uniqueness

In addition to verifiability, Juels, Catalano and Jakobsson [8], as well as Delaune, Kremer and Ryan [12], put forward the notion of *tally uniqueness*. Tally uniqueness of a voting protocol ensures that the tally of an election is unique. In other words, two different tallies $\rho \neq \rho'$ can not be accepted by the verification algorithm, even if all the players in the system are malicious.

More formally, the goal of the adversary against tally uniqueness is to output a public key $\mathbf{pk}$, that contains a list of public credentials, a bulletin board BB, and two tallies $\rho \neq \rho'$, and corresponding proofs of valid tabulation Π and Π', such that both pass verification, i.e. $\mathsf{Verify}(\mathsf{BB}, \rho, \Pi) = \mathsf{Verify}(\mathsf{BB}, \rho', \Pi') = 1$. A voting protocol $\mathcal{V}$ has *tally uniqueness* if every PPT adversary $\mathcal{A}$ has a negligible advantage in this game.

Intuitively, verifiability ensures that the tally corresponds to a plausible instantiations of the players (onto property) while tally uniqueness ensures that, given a tally, there is at most one plausible instantiation (one-to-one property).

4 Sufficient Conditions for Verifiability

In this section we identify sufficient conditions for (individual and universal) verifiability in single-pass voting protocols. In the first place, Section 4.1, we define a property for voting protocols, that we call *accuracy*, and we show that it implies weak verifiability. As explained in the introduction, weak verifiability is not a completely satisfactory property, but it is the highest verifiability level that can be achieved in remote voting systems where the only the bulletin board authenticates voters and therefore it can easily stuff itself. This is notably the case for Helios [9]. Nevertheless, we give in Section 4.3 a generic construction that transforms a voting protocol that has weak verifiability, into a voting protocol that has strong verifiability, namely it is verifiable under the weaker trust assumption that the registrar and the board are *not simultaneously* dishonest.

4.1 Accuracy

We introduce a property for voting protocols that is called *accuracy*. We say that a voting protocol $\mathcal{V}$ has *accuracy* (equivalently it is accurate) if for any ballot b it holds with overwhelming probability that

1. $(\,\mathsf{Validate}(b) = 1 \;\wedge\; \mathsf{Verify}(\{b\}, \rho_b, \Pi_b) = 1\,) \implies \rho_b = \rho(v_b)$ for some $v_b \in \mathbb{V}$
2. $\mathsf{Verify}\,(\mathsf{BB}, \mathsf{Tally}(\mathsf{BB}, \mathbf{sk})) = 1$ for any bulletin board BB

Condition 1 reflects the natural requirement that even a dishonest ballot that passes the validity test corresponds to an admissible vote. In Helios-like protocols, this is typically ensured by requiring the voter to produce a proof that the encrypted vote belongs to $\mathbb{V}$. Condition 2 guarantees that the proof produced by a faithful run of the tally procedure passes the verification test. In practice, this property usually holds by design.

4.2 A Sufficient Condition for Weak Verifiability

We show that correctness (Section 2.2), accuracy (Section 4.1) and tally uniqueness (Section 3.3) suffice to ensure weak verifiability against a dishonest tallying authority. Since these properties are simple and easy to check, this result may often ease the proof of verifiability. We illustrate this fact by using these criteria to give in Section 5 a simple proof that Helios-BPW is weakly verifiable.

Theorem 1. *Let $\mathcal{V}$ be a correct, accurate and tally unique voting protocol that admits partial tallying. Then $\mathcal{V}$ satisfies weak verifiability.*

The proof is given in the full version [18].

Signature Schemes with Verification Uniqueness. We aim at designing a generic construction that provides strong verifiability. Our construction relies on an existentially-unforgeable (EUF-CMA) signature scheme as a building block, whose syntax and properties are given next.

Definition 1 (Signature scheme). *A signature scheme consists of three algorithms $\mathcal{S} = (\mathsf{SKey}, \mathsf{Sign}, \mathsf{SVerify})$, such that*

- $\mathsf{SKey}(1^\lambda)$ *outputs a pair of verification/signing keys* $(\mathsf{upk}, \mathsf{usk})$.
- $\mathsf{Sign}(\mathsf{usk}, m)$ *on inputs a signing key* usk *and a message* m *outputs a signature* σ.
- $\mathsf{SVerify}(\mathsf{upk}, m, \sigma)$ *on inputs a verification key* upk*, a message* m *and a string* σ*, outputs 0/1, meaning invalid/valid signature.*

A signature scheme must satisfy correctness*, namely* $\mathsf{SVerify}(\mathsf{upk}, m, \mathsf{Sign}(\mathsf{usk}, m)) = 1$ *with overwhelming probability, where* $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{SKey}(1^\lambda)$.

We further need to control the behaviour of the signature scheme when keys are (dishonestly) chosen outside the expected range. More precisely, we need to ensure that the output of $\mathsf{SVerify}(\mathsf{upk}, m, \sigma)$ is deterministic, even for inputs outside the corresponding domains. We call this *verification uniqueness.*

4.3 A Sufficient Condition for Strong Verifiability

We provide a generic construction that protects any voting scheme that has weak verifiability, that is assuming that the bulletin board and registrar are *both* honest, into a voting scheme that has string verifiability, that is under the weaker assumption that board and registrar are *not simultaneously* dishonest.

Let $\mathcal{V} = (\mathsf{Setup}', \mathsf{Credential}', \mathsf{Vote}', \mathsf{VerifyVote}', \mathsf{Validate}', \mathsf{Box}', \mathsf{Tally}', \mathsf{Verify}')$ be a voting protocol, possibly without credentials, like Helios. Our generic construction transforms $\mathcal{V}$ into $\mathcal{V}^{\mathsf{cred}}$ as follows. We first require the registration authority to create a public/secret credential pair $(\mathsf{upk}, \mathsf{usk})$ for each voter. Each key pair corresponds to a *credential* needed to cast a vote. The association between credentials and voters does not need to be publicly known and only the unordered list of verification keys (the public credentials) is published. In the resulting voting scheme $\mathcal{V}^{\mathsf{cred}}$, every player acts as in $\mathcal{V}$ except that now, each voter further signs his/her ballot with his/her signing key usk. Moreover, the bulletin board, upon receiving a ballot, performs the usual checks and further verifies the signature (that should correspond to one of the official verification keys). The board also needs to maintain an internal state st that links successful voters' authentications with successful signature verifications, i.e. it keeps links (id, upk_{id}). This is needed to prevent a dishonest voter id', who has gained knowledge of several secret credentials $\mathsf{usk}_1, \ldots, \mathsf{usk}_t$, from stuffing/overriding the board with ballots containing the corresponding public credentials $\mathsf{upk}_1, \ldots, \mathsf{upk}_t$. We call this a *multiple impersonation attack*. Our generic transformation is summarized in Figure 2.

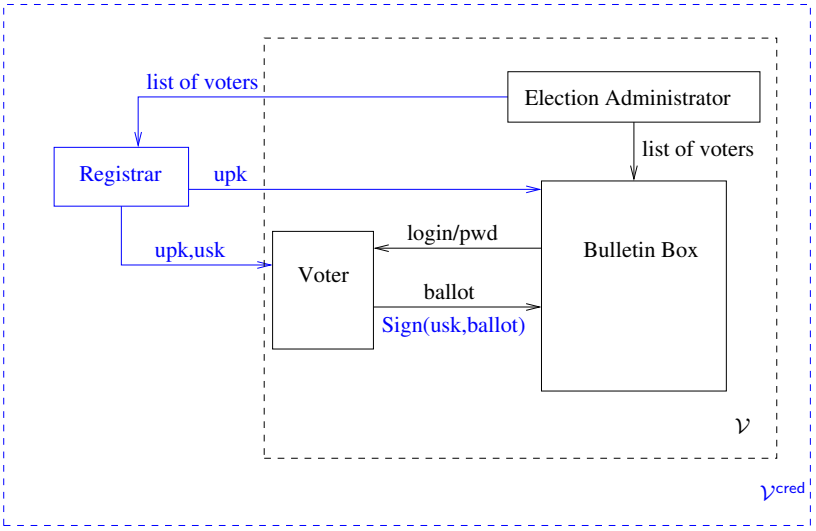

Fig. 2. Generic construction for strong verifiability

Formally, let $\mathcal{S} = (\mathsf{SKey}, \mathsf{Sign}, \mathsf{SVerify})$ be a signature scheme. Let us consider $\mathcal{V}^{\mathsf{cred}} = (\mathsf{Setup}, \mathsf{Credential}, \mathsf{Vote}, \mathsf{Validate}, \mathsf{Box}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{Verify})$ the voting protocol with credentials obtained from $\mathcal{V}$ and $\mathcal{S}$ as follows:

$\mathsf{Setup}(1^\lambda)$ runs $(\mathbf{pk}', \mathbf{sk}') \leftarrow \mathsf{Setup}'(1^\lambda)$ and sets $\mathbf{pk} \leftarrow (\mathbf{pk}', L), \mathbf{sk} \leftarrow \mathbf{sk}'$, where L is a list initialized to empty that is defined below. Let us recall that $\mathbf{pk}'$ potentially contains a list L' of public credentials inherited from $\mathcal{V}'$. Returns $(\mathbf{pk}, \mathbf{sk})$. We say that L is *ill-formed* if $|L| > \tau$, (*i.e.* there are more public credentials than eligible voters) or if L has repeated elements.

$\mathsf{Credential}(1^\lambda, id)$ is run by the registrar and computes $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{SKey}(1^\lambda)$; the bulletin board computes $(\mathsf{upk}', \mathsf{usk}') \leftarrow \mathsf{Credential}'(1^\lambda, id)$. The list L is updated as $L \leftarrow L \cup \{\mathsf{upk}\}$. Next, $\mathbf{upk} \leftarrow (\mathsf{upk}, \mathsf{upk}')$ and $\mathbf{usk} \leftarrow (\mathsf{usk}, \mathsf{usk}')$ are returned.

$\mathsf{Vote}(id, \mathbf{upk}, \mathbf{usk}, v)$ runs $\alpha \leftarrow \mathsf{Vote}'(id, \mathsf{upk}', \mathsf{usk}', v)$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{usk}, \alpha)$ and returns a ballot $b \leftarrow (\mathsf{upk}, \alpha, \sigma)$, which is sent to the bulletin board through an authenticated channel[2].

$\mathsf{Validate}(b)$ parses $b = (\mathsf{upk}, \alpha, \sigma)$. If $\mathsf{SVerify}(\mathsf{upk}, \alpha, \sigma) \neq 1$ outputs 0. Else, outputs $\mathsf{Validate}'(\alpha)$.

$\mathsf{Box}(\mathsf{BB}, b)$ parses $b = (\mathsf{upk}, \alpha, \sigma)$ after a successful authentication, by voter id with credentials $(\mathsf{upk}', \mathsf{usk}')$, to the bulletin board. BB is let unchanged if $\mathsf{upk} \notin L$, or if $\mathsf{Validate}(b)$ rejects. Next (1) if an entry of the form $(id, *)$ or $(*, \mathsf{upk})$ exists in its local state st, then: (1.a) if $(id, \mathsf{upk}) \in st$ and $\alpha \in \mathsf{Box}'(\mathsf{BB}', \alpha)$ (BB' is updated with α), then removes any ballot in BB containing upk, updates $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{b\}$, and returns BB; (1.b) else, returns BB. Otherwise, (2) adds (id, upk) to st, and (2.a) if $\alpha \in \mathsf{Box}'(\mathsf{BB}', \alpha)$, adds b to BB, and returns BB; else (2.b) returns BB. The checks in Steps (1) and (2) are performed to prevent multiple impersonation attacks.

$\mathsf{VerifyVote}(\mathsf{BB}, id, \mathbf{upk}, \mathbf{usk}, b)$ verifies that the ballot b appears in BB. Intuitively, this check should be done by voters when the voting phase is over. If $b = (\mathsf{upk}, \alpha, \sigma) \in \mathsf{BB}$, then outputs $\mathsf{VerifyVote}'(\mathsf{BB}', id, \mathsf{upk}', \mathsf{usk}', \alpha)$. Otherwise, outputs 0.

$\mathsf{Tally}(\mathsf{BB}, \mathbf{sk})$ returns $\rho :=\perp$ and $\Pi := \emptyset$ if L is not well-formed. Else, checks next whether BB is well-formed. We say BB is well-formed if: every upk in BB appears only once; every upk in BB appears in L; $\mathsf{Validate}(b) = 1$ for every $b \in \mathsf{BB}$. If any of these checks fails (meaning that the bulletin board cheated) the trustee outputs $\rho :=\perp$ and $\Pi := \emptyset$. Else the trustee runs $\mathsf{Tally}'(\mathsf{BB}', \mathbf{sk})$, where $\mathsf{BB}' = \{\alpha_1, \ldots, \alpha_\tau\}$ if $\mathsf{BB} = \{(\mathsf{upk}_1, \alpha_1, \sigma_1), \ldots, (\mathsf{upk}_\tau, \alpha_\tau, \sigma_\tau)\}$.

$\mathsf{Verify}(\mathsf{BB}, \rho, \Pi)$ starts by checking whether L and BB are well-formed. If not, outputs 1 if $\rho =\perp$; else it outputs 0. Else, runs $\mathsf{Verify}'(\mathsf{BB}', \rho, \Pi)$, where $\mathsf{BB}' = \{\alpha_1, \ldots, \alpha_\tau\}$ if $\mathsf{BB} = \{(\mathsf{upk}_1, \alpha_1, \sigma_1), \ldots, (\mathsf{upk}_\tau, \alpha_\tau, \sigma_\tau)\}$.

From Weak to Strong Verifiability. Our generic construction converts a weakly verifiable voting scheme into a strongly verifiable voting scheme.

Theorem 2. *Let $\mathcal{V}$ be a voting protocol that satisfies weak verifiability, admits partial tallying and satisfies tally uniqueness. Let $\mathcal{S}$ be an existentially unforgeable signature scheme. Then $\mathcal{V}^{\mathsf{cred}}$ satisfies strong verifiability.*

Proof. It is a consequence of Lemma 1 and Lemma 2 below.

Lemma 1. *Let $\mathcal{V}$ satisfy weak verifiability and tally uniqueness. Let $\mathcal{S}$ be an existentially unforgeable signature scheme. Then $\mathcal{V}^{\mathsf{cred}}$ has verifiability against a dishonest bulletin board.*

[2] This channel is built around the credential information $(id, \mathsf{upk}', \mathsf{usk}')$.

This lemma is proven by showing that any adversary against the verifiability of $\mathcal{V}^{\mathsf{cred}}$, controlling the bulletin board, is "as powerful" as any adversary against the weak verifiability of $\mathcal{V}$, unless it can break the existential unforgeability of the signature scheme $\mathcal{S}$. The proof is given in the full version [18].

Lemma 2. *Let $\mathcal{V}$ be weakly verifiable and tally unique. Then $\mathcal{V}^{\mathsf{cred}}$ has verifiability against a dishonest registrar.*

Note that Lemma 2 relies on the weak verifiability of the voting scheme. Indeed, if the registrar is dishonest, it has all the credentials. Therefore only the bulletin board may prevent him from stuffing the box. Typically, weakly verifiable schemes assume an authenticated channel between the voters and the box, e.g. using some password-based authentication mechanism. This simple proof is given in the full version [18].

Theorem 3. *If $\mathcal{V}$ satisfies tally uniqueness and $\mathcal{S}$ satisfies verification uniqueness, then $\mathcal{V}^{\mathsf{cred}}$ preserves tally uniqueness.*

Our transformation also preserves ballot privacy. Intuitively, this is due to the fact that our transformation of the original protocol does not significantly change the behaviour of the underlying voting scheme. In particular, every valid ballot produced by our transformed voting scheme corresponds to a valid ballot in the original voting scheme, and viceversa. In the full version of this work we give a proof of ballot privacy using the game-based game definition from [7]. The reduction is straightforward and there are no technical difficulties involved.

Theorem 4. *If $\mathcal{V}$ satisfies privacy then $\mathcal{V}^{\mathsf{cred}}$ satisfies privacy.*

5 Helios-C : Helios with Credentials

In this section we modify the design of Helios 4.0 voting system [9]. Actually, the current version does not ensure ballot privacy due to the fact that dishonest voters may duplicate ballots [5]. We therefore consider a slight modification of Helios 4.0 that includes weeding of duplicate ballots and that has been proved secure w.r.t. ballot privacy [7]. We aim at achieving (individual and universal) verifiability under a weaker trust assumption. Our modification consists in adding (verifiable) credentials to prevent ballot stuffing. We name it Helios-C, as a shortening for Helios with Credentials. For readability, we describe Helios for a single choice election (voters may simply vote 0 or 1). It can be easily generalized to elections with several candidates. We assume an authenticated channel between each voter and the bulletin board. This is typically realized in Helios through password-based authentication.

We use the ElGamal [19] IND-CPA cryptosystem $\mathcal{D} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ in a given group $\mathbb{G}$ where the Decisional Diffie-Hellman assumption holds; the Schnorr signature scheme $\mathcal{S} = (\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ [20] over the group $\mathbb{G}$; the NIZK proof system [21, 22] $\mathsf{DisjProof}_H(g, \mathsf{pk}, R, S)$ to prove in zero-knowledge that (R, S) encrypts g^0 or g^1 (with proof builder $\mathsf{DisjProve}$ and proof verifier $\mathsf{DisjVerify}$); and the NIZK proof system [21] $\mathsf{EqDl}_G(g, R, \mathsf{vk}, c)$ to prove in zero-knowledge that $\log_g \mathsf{vk} =$

$\log_R c$ for $g, R, \mathsf{vk}, c \in \mathbb{G}$ (with proof builder PrEq and proof verifier $\mathsf{VerifyEq}$). H and G are hash functions mapping to $\mathbb{Z}_q$.

Formally, Helios-C consists of eight algorithms $\mathcal{V}^{\mathsf{heliosc}} = (\mathsf{Setup}, \mathsf{Credential}, \mathsf{Vote}, \mathsf{Validate}, \mathsf{VerifyVote}, \mathsf{Box}, \mathsf{Tally}, \mathsf{Verify})$ defined below:

$\mathsf{Setup}(1^\lambda)$ chooses $\mathbb{G}$ a cyclic group of order q and $g \in \mathbb{G}$ a generator. It randomly chooses $\mathsf{sk} \xleftarrow{R} \mathbb{Z}_q$ and sets $\mathsf{pk} = g^{\mathsf{sk}}$. Hash functions $G, H : \{0,1\}^\star \to \mathbb{Z}_q$ are chosen. It outputs $\mathbf{pk} \leftarrow (\mathbb{G}, q, \mathsf{pk}, L, G, H, \mathbb{V} = \{0,1\})$, the public key of the election and $\mathbf{sk} = (\mathbf{pk}, \mathsf{sk})$, with L initialized as the empty set.

$\mathsf{Credential}(1^\lambda, id, L)$ generates a signing key pair for each voter. It runs $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$. It adds upk to L and outputs $(\mathsf{upk}, \mathsf{usk})$.

$\mathsf{Vote}\,(id, \mathsf{upk}, \mathsf{usk}, v)$ it is used by a voter of identity id with credentials $(\mathsf{upk}, \mathsf{usk})$ to create a ballot b corresponding to vote v as follows:

(1) Encrypts $v \in \{0,1\}$ as $C = \mathsf{Enc}(\mathsf{pk}, g^v) = (R, S)$. Computes a proof $\pi = \mathsf{DisjProve}_H(g, \mathsf{pk}, R, S, r)$ showing that the encrypted vote is 0 or 1.

(2) Computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{usk}, (C, \pi))$, namely a signature on the ciphertext and its proof. The ballot is defined as $b = (\mathsf{upk}, (C, \pi), \sigma)$.

(3) The voter submits the ballot b by authenticating itself to the bulletin board.

$\mathsf{Validate}(b)$ checks that the ballot is *valid*, that is, that all proofs are correct. Formally, it parses the ballot b as $(\mathsf{upk}, (C, \pi), \sigma)$. It then checks whether: (1) $\mathsf{upk} \in L$; (2) $\mathsf{DisjVerify}_H(g, \mathsf{pk}, C, \pi) = 1$; (4) $\mathsf{SVerify}(\mathsf{upk}, \sigma, (C, \pi))$ accepts. If any step fails, it returns 0; else it returns 1.

$\mathsf{VerifyVote}(id, \mathsf{upk}, \mathsf{usk}, b)$ returns the value of the test $b \in \mathsf{BB}$.

$\mathsf{Box}(\mathsf{BB}, b)$ parses $b = (\mathsf{upk}, (C, \pi), \sigma)$ after a successful authentication from a voter id. BB is let unchanged if $\mathsf{upk} \notin L$, or $\mathsf{Validate}(b)$ rejects or C appears previously in BB. Next, (1) if an entry of the form $(id, *)$ or $(*, \mathsf{upk})$ exists in its local state st, then: (1.a) if $(id, \mathsf{upk}) \in st$, removes any previous ballot in BB containing upk, updates $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{b\}$ and returns BB; (1.b) else, returns BB. Otherwise, (2) adds (id, upk) to st, updates $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{b\}$ and returns BB.

$\mathsf{Tally}(\mathsf{BB}, \mathbf{sk})$ consists of the following steps:

(1) Runs $\mathsf{Validate}(b)$ for every $b \in \mathsf{BB}$. Outputs $\rho = \perp$ and $\Pi = \emptyset$ if any such b is rejected.

(2) Parses each ballot $b \in \mathsf{BB}$ as $(\mathsf{upk}_b, (C_b, \pi_b), \sigma_b)$.

(3) Checks whether upk_b appears in a previous entry in BB or whether $\mathsf{upk}_b \notin L$. If so, outputs $\rho = \perp$ and $\Pi = \emptyset$. Else,

(4) Computes the result ciphertext $C_\Sigma = (R_\Sigma, S_\Sigma) = (\prod_{b \in \mathsf{BB}} R_b, \prod_{b \in \mathsf{BB}} S_b)$, where $C_b = (R_b, S_b)$. This of course relies on the homomorphic property of the El Gamal encryption scheme.

(5) Computes $g^\rho \leftarrow S_\Sigma \cdot (R_\Sigma)^{-\mathsf{sk}}$. Then ρ to be published is obtained from g^ρ in time $\sqrt{\tau}$ for ρ lying in the interval $[0, \tau]$ and τ equals the number of legitimate voters.

(6) Finally $\Pi := \mathsf{PrEq}_G\left(g, \mathsf{pk}, R_\Sigma, S_\Sigma \cdot (g^\rho)^{-1}, \mathsf{sk}\right)$.

$\mathsf{Verify}(\mathsf{BB}, \rho, \Pi)$

(1) Performs the checks (1-3) done in Tally. If any of the checks fails, then returns 0 unless the result is itself $\perp$, in which case outputs 1. Else,
(2) Computes the result ciphertext $(R_\Sigma, S_\Sigma) = \left(\prod_{b\in\mathsf{BB}} R_b, \prod_{b\in\mathsf{BB}} S_b\right)$.
(3) Returns the output of $\mathsf{VerifyEq}_G\left(g, \mathsf{pk}, R_\Sigma, S_\Sigma \cdot (g^\rho)^{-1}, \Pi\right)$.

Theorem 5. *Helios-C has tally uniqueness, strong verifiability and ballot privacy under the Decisional Diffie-Hellman assumption in the Random Oracle Model.*

Since Helios-C = Helios-BPW$^{\mathsf{cred}}$ and the Schnorr signature scheme is EUF-CMA in the Random Oracle Model under the Discrete Logarithm assumption in $\mathbb{G}$, Theorem 2 (Section 4.3) allows to deduce the strong verifiability of Helios-C from the weak verifiability of Helios-BPW. Finally, since Helios-BPW has ballot privacy under the DDH assumption in the Random Oracle Model (Theorem 3 in [7]), then Helios-C has ballot privacy under the same assumptions.

Theorem 6. *Helios-BPW is weakly verifiable under the Discrete Logarithm assumption in the Random Oracle Model.*

Proof. We need to show that Helios-BPW is correct, accurate and has tally uniqueness thanks to Theorem 1. We omit the proof of correctness for Helios-BPW since it easily follows from the correctness of the involved primitives, i.e. the ElGamal cryptosystem, Schnorr signature and NIZKs.

Let us show that Helios-BPW has tally uniqueness, where Helios-BPW = $(\mathsf{Setup}', \mathsf{Vote}', \mathsf{Validate}', \mathsf{VerifyVote}', \mathsf{Box}', \mathsf{Tally}, ' \mathsf{Verify}')$. The output of Verify' is determined by the outputs of the verification tests of the NIZK systems $\mathsf{DisjProof}_H$ and EqDl_G, which constitute proof of memberships to the corresponding languages with negligible error probability, and hence the output of Verify' is unique on his inputs.

With respect to the accuracy of Helios-BPW, we need to show that for any ballot b it holds that if $\mathsf{Validate}'(b) = 1$ and $\mathsf{Verify}'(\{b\}, \rho_b, \Pi_b) = 1$, then $\rho_b = \rho(v_b)$ for some $v_b \in \mathbb{V}$. Let $\alpha = (C, \pi)$ be such that $\mathsf{DisjVerify}(g, \mathsf{pk}, C, \pi) = 1$. Since $\mathsf{DisjProof}_H$ is a NIZK obtained by applying Fiat-Shamir to a Σ-protocol [23], then $\mathsf{DisjProof}_H$ is a proof that $(g, \mathsf{pk}, R_b, S_b) \in \mathcal{L}_{\mathsf{EqDl}}$ or $(g, \mathsf{pk}, R_b, S_b \cdot g^{-1}) \in \mathcal{L}_{\mathsf{EqDl}}$ with soundness error $1/q$. In other words, if $\mathsf{Validate}'(b) = 1$ and $\mathsf{Verify}'(\{b\}, \rho_b, \Pi_b) = 1$, then $v_b \in \{0, 1\}$ with overwhelming probability. This proves accuracy of Helios-BPW. □

6 Implementation

We have implemented a proof of concept of Helios-C, openly accessible at [24], and tested it in a mock election in our lab.

In Helios-C credentials are generated by a third-party provider and sent to the voters by snail mail. Clearly, it would be cumbersome for voters to copy their signature key by typing it. We used a trick that consists in sending only the random seed used for generating the key, which can be encoded in about 12-15 alphanumeric characters depending on the desired entropy. It is expected that this seed is used by the provider to add the generated public key to L, then sent (as a password) to its rightful recipient and immediately destroyed.

candidates	2	5	10	20	30	50
enc+proofs	600	1197	2138	4059	6061	9617
sign	196	215	248	301	358	484
sig verif	< 10	< 10	< 10	< 10	< 10	< 10
ballot verif	110	210	390	720	1070	1730

Fig. 3. Overhead in miliseconds induced by adding credentials to Helios

Our variant of Helios requires voters to additionally sign their ballots. Table 3 shows the overhead induced by the signature, for various numbers of candidates (from 2 to 50). The two first lines are timings on the client side: the first one indicates the time needed by the voter's browser to form the ballot (without signature) while the second line indicates the computation time for signing. The third and fourth lines indicate the computation time on the server side for performing the verification tests (well-formedness of the ballot, validity of the proofs of knowledge and validity of the signature). Since the ballot includes the public key of the voter, the server simply needs to verify one signature for each ballot and to verify that the public keys indeed belongs to the set of authorized keys, which can be done in logarithmic time. We use a 256-bit multiplicative subgroup of a 2048-bit prime field for ElGamal and Schnorr operations. The figures have been obtained on a computer with an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz, running Firefox 18. Unsurprisingly, the overhead of the signature is small compared to the computation time of the whole ballot.

We have tested our implementation in a mock election in June 2013, among approximately 30 voters. The result of the election and in particular all its public data (including ballots) can be found at [24].

In practice, it is also needed to provide a password/credential recovery procedure in case voters lose their credentials. In case revoting is authorized, we further assume that the registrar keeps the link between users and public credentials during the election so that the old (lost) credential can be erased from the authorized list.

7 Conclusion

We have presented a generic construction that enforces strong verifiability. Applied to Helios, the resulting system Helios-C prevents ballot stuffing, still retaining the simplicity of Helios, as demonstrated by our test election, under the trust assumption that registrar and bulletin board are *not simultaneously* dishonest. For simplicity, we have presented our framework for a single vote (yes/no vote) and for a single trustee. All our results can be easily extended to multiple candidates elections and multiple trustees, possibly with threshold decryption as described in [25].

We would like to point out a more appealing variant of our transformation from a theoretical point of view. In this variant, voters generate their individual credentials (i.e. a signing key pair) by themselves. Thus a malicious registrar cannot sign on behalf of honest users, as it would only be responsible of registering credentials for eligible voters. We think, however, that letting the registrar generate credentials on behalf of voters, as we do in Helios-C, is a more practical choice: most voters will not have the

required knowledge to perform the critical procedure of generating credentials with a minimum of security guarantees.

Even if most ballot counting functions admit partial tallying, especially for practical counting functions, some functions do not admit partial tallying, like the majority function. As future work, we plan to investigate whether we can devise a definition of verifiability for schemes that do not admit partial tallying.

Strong verifiability of Helios-C assumes that either the registration authority or the ballot box is honest. We could further thresholdize the registration authority, by distributing each credential among several registrars. We plan to explore the possibility to go further and design a (practical) voting scheme that offers verifiability without any trust assumption (like vote by hand-rising), and ballot privacy under some trust assumptions, like the fact that some of the authorities are honest.

References

1. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: Analysis of real-world use of Helios. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (2009)
2. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
3. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: Proceedings of the Second Usenix/ACCURATE Electronic Voting Technology Workshop (2007)
4. International association for cryptologic research, Elections page at `http://www.iacr.org/elections/`
5. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. In: CSF, pp. 297–311. IEEE Computer Society (2011)
6. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting Helios for provable ballot secrecy. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 335–354. Springer, Heidelberg (2011)
7. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
8. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 37–63. Springer, Heidelberg (2010)
9. Adida, B., de Marneffe, O., Pereira, O.: Helios voting system, `http://www.heliosvoting.org`
10. Pinault, T., Courtade, P.: E-voting at expatriates' MPs elections in France. In: Kripp, M.J., Volkamer, M., Grimm, R. (eds.) Electronic Voting. LNI, vol. 205, pp. 189–195. GI (2012)
11. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
12. Delaune, S., Kremer, S., Ryan, M.D., Steel, G.: A formal analysis of authentication in the TPM. In: Degano, P., Etalle, S., Guttman, J. (eds.) FAST 2010. LNCS, vol. 6561, pp. 111–125. Springer, Heidelberg (2011)
13. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 526–535. ACM (2010)

14. Küsters, R., Truderung, T., Vogt, A.: Verifiability, privacy, and coercion-resistance: New insights from a case study. In: IEEE Symposium on Security and Privacy, pp. 538–553. IEEE Computer Society (2011)
15. Küsters, R., Truderung, T., Vogt, A.: Clash attacks on the verifiability of e-voting systems. In: IEEE Symposium on Security and Privacy, pp. 395–409. IEEE Computer Society (2012)
16. Groth, J.: Evaluating security of voting schemes in the universal composability framework. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 46–60. Springer, Heidelberg (2004)
17. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., di Vimercati, S.D.C., Dingledine, R. (eds.) WPES, pp. 61–70. ACM (2005)
18. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election verifiability for Helios under weaker trust assumptions. HAL - INRIA Archive Ouverte/Open Archive, Research Report RR-8855 (2014), http://hal.inria.fr/hal-01011294
19. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
20. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
21. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) Advances in Cryptology - CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
22. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) Advances in Cryptology - CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
23. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography. Springer (2010)
24. Glondu, S.: Helios with Credentials: Proof of concept and mock election results, http://stephane.glondu.net/helios/
25. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed ElGamal à la Pedersen: Application to Helios. In: Sadeghi, A.R., Foresti, S. (eds.) WPES, pp. 131–142. ACM (2013)