D4.3 Results on a real life case study: Helios 2.0

Véronique Cortier and Steve Kremer

January 16, 2012

The results presented in this report have been obtained by David Bernhard, Véronique Cortier, Steve Kremer, Olivier Pereira, Mark Ryan, Ben Smyth, and Bogdan Warinschi.

Abstract

Helios 2.0 is an open-source web-based end-to-end verifiable electronic voting system, suitable for use in low-coercion environments. In this report, we present an attack against ballot secrecy, a fix and a cryptographic proof of the corrected version of Helios.

1 Introduction

Helios is an open-source web-based electronic voting system. Helios is particularly significant due to its real-world deployment: the International Association of Cryptologic Research used Helios to elect its board members [BVQ10], following a successful trial in a non-binding poll [HBH10]; the Catholic University of Louvain adopted the system to elect the university president [AMPQ09]; and Princeton University used Helios to elect the student vice president [Pri10].

The scheme is claimed to satisfy ballot secrecy [AMPQ09], but the nature of remote voting makes the possibility of satisfying stronger privacy properties difficult, and Helios does not satisfy receipt freeness nor coercion resistance. In addition to ballot secrecy, the system aims at providing end-to-end verifiability.

This report summarizes the results obtained on the Helios protocol during the AVOTÉ project.

- First, we have found an attack against ballot secrecy. This attack has been acknowledged by the authors of Helios and will be corrected in the next releases. We have proposed a fixed and proved its security in formal models. This result has been presented at CSF 2011 [CS11].

- Second, we have shown ballot secrecy of the fixed version in cryptographic models. These models offer better security guarantees but the proof are of course more demanding. We have investigated an abstract voting scheme (of which the revised Helios is an instantiation) built from an arbitrary encryption scheme with certain functional properties. We prove, generically, that whenever this encryption scheme falls in the class of *voting-friendly* schemes that we define, the resulting voting scheme provably satisfies ballot privacy.

We explain how our general result yields cryptographic security guarantees for the revised version of Helios (albeit from non-standard assumptions).

Furthermore, we show (by giving two distinct constructions) that it is possible to construct voting-friendly encryption, and therefore voting schemes, using only standard cryptographic tools.

This result has been presented at ESORICS 2011 [BCP+11].

- Third, we have studied end-to-end verifiability of HELIOS, which was one of the main goals of the protocol. We have shown that the protocol indeed respects individual and universal verifiability. We also identified a third form of verifiability that we called eligibility verifiability: anyone observer can verify that the outcomes consists of votes of eligible voters (and only one vote for each eligible voter). This property is sometimes included in universal verifiability, but most often neglected. We have shown that HELIOS does not verify this property and hence may be subject to ballot stuffing by election organizers.

  These results have been published at ESORICS 2010 [KRS10].

The three papers are attached to this report.

2 Attack against ballot secrecy

Formal definitions of ballot secrecy have been introduced in the context of the applied pi calculus by Delaune, Kremer & Ryan [KR05, DKR06, DKR09, DKR10] and Backes, Hriţcu & Maffei [BHM08]. These privacy definitions consider two voters $\mathcal{A}$, $\mathcal{B}$ and two candidates t, t'. Ballot secrecy is captured by the assertion that an adversary (controlling arbitrary many dishonest voters) cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. This can be expressed by the following equivalence.

$$\mathcal{A}(t) \mid \mathcal{B}(t') \approx_l \mathcal{A}(t') \mid \mathcal{B}(t)$$

These formal definitions of ballot secrecy have been used by their respective authors to analyse the electronic voting protocols due to: Fujioka, Okamoto & Ohta [FOO92], Okamoto [Oka98], Lee *et al.* [LBD+04], and Juels, Catalano & Jakobsson [JCJ02, JCJ05, JCJ10]. It therefore seems natural to check whether Helios satisfies ballot secrecy as well.

Our analysis of Helios reveals an attack which violates ballot secrecy. The attack exploits the system's lack of ballot independence, and works by replaying a voter's ballot or a variant of it (without knowing the vote contained within that ballot). Replaying a voter's ballot immediately violates ballot secrecy in an election with three voters. For example, consider the electorate Alice, Bob, and Mallory; if Mallory replays Alice's ballot, then Mallory can reveal Alice's vote by observing the election outcome and checking which candidate obtained at least two votes. The practicality of this attack has been demonstrated by violating privacy in a mock election using the current Helios implementation. Furthermore, the vulnerability can be exploited in more realistic settings and, as an illustrative example, we discuss the feasibility of the attack in French legislative elections. This case study suggests there is a plausible threat to ballot secrecy. We have also proposed a variant of the attack which abuses the malleability

of ballots to ensure replayed ballots are distinct: this makes identification of replayed ballots non-trivial (that is, checking for exact duplicates is insufficient).

Nonetheless, we have fixed the Helios protocol by identifying and discarding replayed ballots. We believe this solution is particular well-suited because it maintains Benaloh's principle of ballot casting assurance [Ben06, Ben07] and requires a minimal extension to the Helios code-base. Finally, we have shown that the revised scheme satisfies a formal definition of ballot secrecy using the applied pi calculus.

3 Computational ballot secrecy of the fixed version

The revised scheme has been proved secure in a symbolic model but its security in the stronger, computational sense was not yet assessed. We have started by providing a *computational* security model for ballot privacy. In a sense, our model generalizes and strengthens the model of [KR05, DKR06] where an attacker tries to distinguish when two ballots are swapped. Here, we ask that the adversary cannot detect whether the ballots cast are ballots for votes that the adversary has chosen or not. In doing so, the adversary is allowed to control arbitrarily many players and see the result of the election. Our model uses cryptographic games and thus avoids imposing the more onerous constraints that other definitional styles (in particular simulability) require from protocols.

Next we turned our attention to the revised version of Helios. Our analysis follows a somewhat indirect route: instead of directly analysing the scheme as it has been implemented, we analyze an abstract version of Helios that follows the same basic architecture, but where the concrete primitives are replaced with more abstract versions. Of course, the version we analyze implements the suggested weeding of ballots. We present this abstract scheme as a generic construction of a voting scheme starting from encryption scheme with specific functional and security properties.

Focusing on this more abstract version brings important benefits. Firstly, we pin-down more clearly the requirements that the underlying primitives should satisfy. Specifically, we identify a class of *voting-friendly* encryption schemes which when plugged in our construction yield voting schemes with provable ballot privacy. Roughly speaking, such encryption schemes are IND-CCA2 secure and have what we call a homomorphic embedding (parts of the ciphertexts can be seen as ciphertexts of a homomorphic encryption scheme). Secondly, our analysis applies to all voting schemes obtained as instantiations of our generic construction. Although we analyze and propose constructions which for efficiency reasons resort to random oracles, our generic approach also invites other (non-random oracle based) instantiations.

Next, we have shown how to construct voting-friendly encryption schemes using standard cryptographic tools. We discuss two distinct designs. The first construction starts from an arbitrary (IND-CPA) homomorphic encryption scheme and attaches to its ciphertexts a zero-knowledge proof of knowledge of the plaintext. We refer to this construction as the Enc+PoK construction. Despite its intuitive appeal, we currently do not know how to prove that the above design leads to an IND-CCA2 secure encryption scheme (a proprety demanded by voting-friendliness). We therefore cannot conclude the security of our generic scheme when implemented with an arbitrary Enc+PoK scheme. Nevertheless, an investigation into this construction is important since the instantiation where Enc is the ElGamal scheme and PoK is obtained using the Fiat-Shamir paradigm applied to a Schnorr-like protocol corresponds precisely to the encryption scheme currently used in Helios. The security of this specific

construction has been analyzed in prior work. Tsiounis and Yung [TY98] and Schnorr and Jakobsson [SJ00] demonstrate that the scheme is IND-CCA2 secure, but their proofs rely on highly non-standard assumptions. Nevertheless, in conjunction with the security of our main construction, one can conclude that the current implementation of Helios satisfies ballot privacy based on either the assumption in [TY98] or those of [SJ00].

We then take a closer look at the Enc+PoK construction and revisit a technical reason that prevents an IND-CCA2 security proof, first studied by Shoup and Gennaro [SG98]. Very roughly, the problem is that the knowledge extractor associated to the proof of knowledge may fail if used multiple times since its associated security guarantees are only for constant (or logarithmically many) uses. With this in mind, we note that a security proof is possible if the proof of knowledge has a so called *straight line* extractor [Fis05]. This type of extractor can be reused polynomially many times. In this case, the Enc+PoK construction leads to a voting-friendly encryption scheme, whenever Enc is an arbitrary IND-CPA homomorphic encryption scheme.

The second design uses the well-known Naor-Yung transformation [NY90]. We have shown that if the starting scheme is an arbitrary (IND-CPA) homomorphic encryption scheme then the result of applying the NY transform is a voting-friendly encryption scheme. Applied generically, the transform may lead to non-efficient schemes (one of its components is a simulation-sound zero-knowledge proof of membership [Sah99]). We presented a related construction (where the proof of membership is replaced by a proof of knowledge) which can be efficiently instantiated in the random oracle model. Putting all our results together, we propose adopting an instantiation of Helios where the encryption-friendly scheme is implemented as above. The computational overhead for this scheme is reasonable (and can be further improved through specific optimization) and the scheme comes with the formal guarantees offered by our results.

4 Election verifiability in HELIOS

A major difference between traditional paper based and electronic elections is the lack of transparency of the later. In paper elections it is often possible to observe the whole process from ballot casting to tallying, and to rely on robustness characteristics of the physical world (such as the impossibility of altering the markings on a paper ballot sealed inside a locked ballot box). By comparison, it is not possible to observe the electronic operations performed on data. Computer systems may alter voting records in a way that cannot be detected by either voters or election observers. A voting terminal's software might be infected by malware which could change the entered vote, or even execute a completely different protocol than the one expected.

The concept of *election* or *end-to-end verifiability* that has emerged in the academic literature, *e.g.*, [JCJ02, JCJ05, CRS05, Adi08, Par07, Adi06], aims to address this problem. It should allow voters and election observers to verify, independently of the hardware and software running the election, that votes have been recorded, tallied and declared correctly. One generally distinguishes two aspects of verifiability.

- *Individual verifiability:* a voter can check that her own ballot is included in the election's bulletin board.
- *Universal verifiability:* anyone can check that the election outcome corresponds to the ballots published on the bulletin board.

We identify another aspect that is sometimes included in universal verifiability.

- *Eligibility verifiability:* anyone can check that each vote in the election outcome was cast by a registered voter and there is at most one vote per voter.

We explicitly distinguish eligibility verifiability as a distinct property.

We present a definition of election verifiability which captures the three desirable aspects. We model voting protocols in the applied pi calculus and formalise verifiability as a triple of boolean tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$ which are required to satisfy several conditions on all possible executions of the protocol. Φ^{IV} is intended to be checked by the individual voter who instantiates the test with her private information (*e.g.*, her vote and data derived during the execution of the protocol) and the public information available on the bulletin board. Φ^{UV} and Φ^{EV} can be checked by any external observer and only rely on public information, *i.e.*, the contents of the bulletin board.

The consideration of eligibility verifiability is particularly interesting as it provides an assurance that the election outcome corresponds to votes legitimately cast and hence provides a mechanism to detect ballot stuffing. We note that this property has been largely neglected in previous work and our earlier work [SRKK10] only provided limited scope for.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. As it is not reasonable to assume voting systems behave correctly we only model the parts of the protocol that we need to trust for the purpose of verifiability; all the remaining parts of the system will be controlled by the adversarial environment. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter.* In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and protocols often need to trust some parts of the voting software or some administrators. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or can be executed in a distributed manner amongst several different election officials. For instance, in Helios 2.0 [Adi08], the ballot construction can be audited using a cast-or-audit mechanism. Whether trust assumptions are reasonable depends on the context of the given election, but our work makes them explicit.

Tests Φ^{IV}, Φ^{UV} and Φ^{EV} are assumed to be verified in a trusted environment (if a test is checked by malicious software that always evaluates the test to hold, it is useless). However, the verification of these tests, unlike the election, can be repeated on different machines, using different software, provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed in [Adi06, Chapter 5].

We have shown that Helios is individual and universal verifiable. For this to be the case it relies however on some trust assumptions. The parts of the system that are not verifiable are:

- The script that constructs the ballot. Although the voter cannot verify it, the trust in this script is motivated by the fact that she is able to audit it.

- The trustees. Although the trustees' behaviour cannot be verified, voters and observers may want to trust them because trust is distributed among them.

We have also shown that Helios 2.0 does not guarantee eligibility verifiability and is vulnerable to ballot stuffing by dishonest administrators.

References

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2006.

[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[AMPQ09] 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.

[BCP+11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot secrecy. In Springer, editor, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, volume 6879 of *Lecture Notes in Computer Science*, 2011.

[Ben06] Josh Benaloh. Simple Verifiable Elections. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[Ben07] Josh Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[BHM08] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08: 21st Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[BVQ10] 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`, Sept 2010.

[CRS05] 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.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*. IEEE Computer Society Press, June 2011.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *CSFW'06: 19th Computer Security Foundations Workshop*, pages 28–42. IEEE Computer Society, 2006.

[DKR09] 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.

[DKR10] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster. 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 289–309. Springer, 2010.

[Fis05] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Proceedings of the 25th annual international cryptology conference on advances in cryptology (CRYPTO '05)*, pages 152–168, 2005.

[FOO92] 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.

[HBH10] 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`, May 2010.

[JCJ02] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[JCJ05] 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.rsa.com/rsalabs/node.asp?id=2860`.

[JCJ10] 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.

[KR05] 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.

[KRS10] 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.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In *ICISC'03: 6th International Conference on Information Security and Cryptology*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on theory of computing (STOC '90)*, 1990.

[Oka98] Tatsuaki Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In *SP'97: 5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 25–35. Springer, 1998.

[Par07] Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl Accord. `http://www.dagstuhlaccord.org/`, 2007.

[Pri10] Princeton University. Princeton election server. `https://princeton-helios.appspot.com/`, 2010.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of th 40th annual symposium on foundations of computer science (FOCS '99)*, pages 543–553, 1999.

[SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems agains chosen-ciphertext attack. In *Advances in Cryptology (Eurocrypt '98)*, volume 1403 of *LNCS*, pages 1–16, 1998.

[SJ00] C.P. Schnorr and M. Jakobsson. Security of signed elgamal encryption. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '00)*, pages 73–89, 2000.

[SRKK10] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, volume 6186 of *LNCS*, pages 165–182. Springer, 2010.

[TY98] Y. Tsiounis and M. Yung. On the security of elgamal-based encryption. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, pages 117–134, 1998.

Attacking and fixing Helios: An analysis of ballot secrecy*

Véronique Cortier[1] and Ben Smyth[2]

[1]Loria, CNRS & INRIA Nancy Grand Est, France
[2]Toshiba Corporation, Kawasaki, Japan

November 10, 2011

Abstract

Helios 2.0 is an open-source web-based end-to-end verifiable electronic voting system, suitable for use in low-coercion environments. In this article, we analyse ballot secrecy and discover a vulnerability which allows an adversary to compromise the privacy of voters. The vulnerability exploits the absence of ballot independence in Helios and works by replaying a voter's ballot or a variant of it, the replayed ballot influences the election outcome, introducing information that can be used to violate privacy. We demonstrated the practicality of the attack by breaking privacy in a mock election using the current Helios implementation. Moreover, the feasibility of an attack is considered in the context of French legislative elections and, based upon our findings, we believe it constitutes a real threat to ballot secrecy in such settings. We present a fix and show that our solution satisfies a formal definition of ballot secrecy using the applied pi calculus. In addition, we present attacks against other electronic voting schemes which do not assure ballot independence; in particular, we show a similar vulnerability in the protocol by Lee *et al.* and demonstrate replay attacks against the schemes by Sako & Kilian and Schoenmakers. Finally, we argue that no general relationships exist between independence and privacy properties.

Keywords. Applied Pi Calculus, Attack, Ballot Independence, Ballot Secrecy, Electronic Voting, Fiat-Shamir Heuristic, Helios, Malleability, Privacy, Replay.

*A preliminary version of this article was presented at CSF'11 [26]. Ben Smyth's work was partly done at Loria, CNRS & INRIA Nancy Grand Est, France.

1 Introduction

Paper-based elections derive security properties from physical characteristics of the real-world. For example, marking a ballot in the isolation of a polling booth and depositing the completed ballot into a locked ballot box provides privacy; the polling booth also ensures that voters cannot be influenced by other votes and the locked ballot box prevents the announcement of early results, thereby ensuring fairness; and the transparency of the whole election process from ballot casting to tallying and the impossibility of altering the markings on a paper ballot sealed inside a locked ballot box gives an assurance of correctness and facilitates verifiability. Replicating these attributes in a digital setting has proven to be difficult and, hence, the provision of secure electronic voting systems is an active research topic.

Informally, *privacy* for electronic voting systems is characterised by the following requirements [49, 30, 9]:

- *Ballot secrecy.* A voter's vote is not revealed to anyone.
- *Receipt freeness.* A voter cannot gain information which can be used to prove, to a coercer, how she voted.
- *Coercion resistance.* A voter cannot collaborate, with a coercer, to gain information which can be used to prove how she voted.

Verifiability includes three properties [47, 62, 50]:

- *Individual verifiability.* A voter can check that her own ballot is published on the election's bulletin board.
- *Universal verifiability.* Anyone can check that all the votes in the election outcome correspond to ballots published on the election's bulletin board.
- *Eligibility verifiability.* Anyone can check that each ballot published on the bulletin board was cast by a registered voter and at most one ballot is tallied per voter.

Finally, *fairness* – summarised by the notion that *all voters are equal* – has not be thoroughly studied, but nonetheless we believe the following aspects are desirable:

- *Ballot independence.* Observing another voter's interaction with the election system does not allow a voter to cast a *related* vote.
- *No early results.* A voter cannot change her vote once partial results are available.
- *Pulling out.* Once partial results are available a voter cannot abort.

The fairness property prohibits the voting system from influencing a voter's vote; more formally, this requires that observation of the voting system (that is,

observing interaction between participants) does not leak information that may affect a voter's decision. The individual, universal and eligibility verifiability properties (also called *end-to-end verifiability* [46, 20, 4, 62, 5]) allow voters and election observers to verify – independently of the hardware and software running the election – that votes have been recorded, tallied and declared correctly. In this article, we analyse ballot secrecy in Helios 2.0 [7].

The Helios protocol. Helios is an open-source web-based electronic voting system. The scheme is claimed to satisfy ballot secrecy [7], but the nature of remote voting makes the possibility of satisfying stronger privacy properties difficult, and Helios does not satisfy receipt freeness nor coercion resistance. In addition to ballot secrecy, the system provides individual and universal verifiability (cf. [50, 78] and [75, Chapter 3] for an analysis of verifiability in Helios). Helios is particularly significant due to its real-world deployment: the International Association of Cryptologic Research used Helios to elect its board members [13], following a successful trial in a non-binding poll [45]; the Catholic University of Louvain adopted the system to elect the university president [7]; and Princeton University used Helios to elect the student vice president [66].

Formal definitions of ballot secrecy have been introduced in the context of the applied pi calculus by Delaune, Kremer & Ryan [49, 30, 31, 32] and Backes, Hriţcu & Maffei [9]. These privacy definitions consider two voters $\mathcal{A}$, $\mathcal{B}$ and two candidates t, t'. Ballot secrecy is captured by the assertion that an adversary (controlling arbitrary many dishonest voters) cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. This can be expressed by the following equivalence.

$$\mathcal{A}(t) \mid \mathcal{B}(t') \approx_l \mathcal{A}(t') \mid \mathcal{B}(t)$$

These formal definitions of ballot secrecy have been used by their respective authors to analyse the electronic voting protocols due to: Fujioka, Okamoto & Ohta [42], Okamoto [59], Lee *et al.* [55], and Juels, Catalano & Jakobsson [46, 47, 48]. It therefore seems natural to check whether Helios satisfies ballot secrecy as well.

Contribution. Our analysis of Helios reveals an attack which violates ballot secrecy. The attack exploits the system's lack of ballot independence, and works by replaying a voter's ballot or a variant of it (without knowing the vote contained within that ballot). Replaying a voter's ballot immediately violates ballot secrecy in an election with three voters. For example, consider the electorate Alice, Bob, and Mallory; if Mallory replays Alice's ballot, then Mallory can reveal Alice's vote by observing the election outcome and checking which candidate obtained at least two votes. The practicality of this attack has been demonstrated by violating privacy in a mock election using the current Helios implementation. Furthermore, the vulnerability can be exploited in more realistic settings and,

as an illustrative example, we discuss the feasibility of the attack in French legislative elections. This case study suggests there is a plausible threat to ballot secrecy. We also propose a variant of the attack which abuses the malleability of ballots to ensure replayed ballots are distinct: this makes identification of replayed ballots non-trivial (that is, checking for exact duplicates is insufficient). Nonetheless, we fix the Helios protocol by identifying and discarding replayed ballots. We believe this solution is particular well-suited because it maintains Benaloh's principle of ballot casting assurance [11, 12] and requires a minimal extension to the Helios code-base. The revised scheme is shown to satisfy a formal definition of ballot secrecy using the applied pi calculus.

In addition, we demonstrate that the absence of ballot independence can be exploited in other electronic voting protocols to violate privacy; in particular, a similar attack is shown against the protocol by Lee *et al.* [55] whereby an adversary replays a voter's ballot or a variant of it, and verbatim replay attacks are demonstrated against two schemes presented at CRYPTO (namely, the protocols due to Sako & Kilian [70] and Schoenmakers [72]). Finally, we present some evidence to support the hypothesis that independence does not imply privacy and vice-versa.

Related work. The attack against Helios that we discover relies upon the lack of ballot independence (in particular, ballots can be replayed). The concept of independence was introduced by Chor *et al.* [21] and the possibility of compromising security properties due to lack of independence has been considered, for example, by [22, 34, 35, 44]. In the context of electronic voting, Gennaro [43] demonstrates that the application of the Fiat-Shamir heuristic in the Sako-Kilian electronic voting protocol [70] violates ballot independence, and Wikström [83, 84] studies non-malleability for mixnets to achieve ballot independence. By comparison, we focus on the violation of ballot secrecy rather than fairness, and exploit the absence of ballot independence to compromise privacy. Similar results have been shown against mixnets [65].

Our attack is also reliant on the voter's ability to cast a ballot as a function of another voter's ballot. In particular, the basic form of our attack applies the identity function, and the variant exploiting malleability performs a permutation on the ballot's internal structure. In related work, Benaloh [10] demonstrates that a simplified version of his voting scheme allows the administrator's private key to be recovered by a voter who constructs their ballot as a function of other voters' ballots.

Estehghari & Desmedt [39] claim to present an attack which undermines privacy and end-to-end verifiability in Helios. However, their attack is dependent on compromising a voter's computer, a vulnerability which is explicitly acknowledged by the Helios specification [7]: *"a specifically targeted virus could surreptitiously change a user's vote and mask all of the verifications performed via the same computer to cover its tracks."* Accordingly, [39] represents an exploration of known vulnerabilities rather than an attack.

Other studies of Helios have also been conducted. In particular, Langer *et*

al. [52, 53] and Volkamer & Grimm [80] study privacy in Helios. Langer *et al.* propose a taxonomy of informal privacy requirements [52, 53, 54] to facilitate a more fine-grained comparison of electronic voting systems; this framework is used to analyse Helios and the authors claim ballot secrecy is satisfied if the adversary only has access to public data [52, 53]. Volkamer & Grimm introduce the *k-resilience* metric [80, 79] to calculate the number of honest participants required for ballot secrecy in particular scenarios; this framework is used to analyse Helios and the authors claim ballot secrecy is satisfied if the software developers are honest and the key holders do not collude [80]. Contrary to these results, we show an attack against privacy. Our work highlights the necessity for rigorous mathematical analysis techniques for security protocols; in particular, we believe the erroneous results reported by Langer *et al.* were due to the use of informal methods, and the approach by Volkamer & Grimm failed because only some particular scenarios were considered.

Delaune, Kremer & Ryan [30, 31] have shown that a variant of the Lee *et al.* protocol satisfies coercion resistance for two honest voters; but, based upon our preliminary results [26], Dreier, Lafourcade & Lakhnech [36] demonstrate an attack against privacy for three voters, when one voter is under the adversary's control[1]. Furthermore, using a stronger definition of coercion resistance, Küsters & Truderung [51] have demonstrated a forced abstention attack; in addition, Küsters & Truderung propose a variant of the scheme by Lee *et al.* which is claimed to satisfy their stronger definition. In this article, we show a new attack against the original Lee *et al.* protocol and show that the revised scheme by Küsters & Truderung might not be secure under reasonable assumptions.

A preliminary version of this work [26] appeared at the 24th Computer Security Foundations Symposium. By comparison, in this article, we provide a more detailed description of our results, generalise our proof of ballot secrecy to a setting with arbitrary many candidates, and include complete proofs. In addition, we show that other electronic voting protocols are vulnerable to our attack and we discuss the link between ballot independence and privacy.

Structure of this article. Section 2 presents the Helios electronic voting scheme. (We remark that this is the first cryptographic description of the Helios protocol in the literature and, hence, is an additional contribution of this article.) Section 3 describes our attack and some variants, in addition to a study of its feasibility in the context of French legislative elections. We propose several solutions for recovering privacy in Section 4 and prove that our adopted solution formally satisfies ballot secrecy in Section 5. Section 6 demonstrates that the absence of ballot independence can be similarly exploited in other electronic voting protocols to violate privacy. Finally, our conclusion appears in Section 7.

[1]The formal model by Dreier, Lafourcade & Lakhnech includes the voter's signature on the signed re-encrypted ciphertext and this is exploited by their attack; by comparison, the model by Delaune, Kremer & Ryan omits this detail and therefore the attack cannot be witnessed.

2 Background: Helios 2.0

We provide a full description of Helios 2.0. This scheme exploits the additive homomorphic [27, 29, 73] and distributed decryption [63, 19] properties of ElGamal [37]. We will recall these cryptographic details before presenting the Helios protocol.

2.1 Additive homomorphic ElGamal

Given cryptographic parameters (p, q, g) and a number $n \in \mathbb{N}$ of trustees, where p and q are large primes such that $q \mid p-1$ and g is a generator of the multiplicative group $\mathbb{Z}_p^*$ of order q, the following operations are defined by ElGamal.

Distributed key generation. Each trustee $i \in n$ selects a private key share $x_i \in_R \mathbb{Z}_q^*$ and computes a public key share $h_i = g^{x_i} \bmod p$. The public key is $h = h_1 \cdot \ldots \cdot h_n \bmod p$.

Encryption. Given a message m and a public key h, select a random nonce $r \in_R \mathbb{Z}_q^*$ and derive the ciphertext $(a, b) = (g^r \bmod p,\ g^m \cdot h^r \bmod p)$.

Re-encryption. Given a ciphertext (a, b) and public key h, select a random nonce $r' \in_R \mathbb{Z}_q^*$ and derive the re-encrypted ciphertext $(a', b') = (a \cdot g^{r'} \bmod p,\ b \cdot h^{r'} \bmod p)$.

Homomorphic addition. Given two ciphertexts (a, b) and (a', b'), the homomorphic addition of plaintexts is computed by multiplication $(a \cdot a' \bmod p,\ b \cdot b' \bmod p)$.

Distributed decryption. Given a ciphertext (a, b), each trustee $i \in n$ computes the partial decryption $k_i = a^{x_i}$. The plaintext $m = \log_g M$ is recovered from $M = b/(k_1 \cdot \ldots \cdot k_n) \bmod p$.

The computation of a discrete logarithm $\log_g M$ is hard in general. However, if M is chosen from a restricted domain, then the complexity is reduced; for example, if M is an integer such that $0 \leq M \leq n$, then the complexity is $O(n)$ by linear search or $O(\sqrt{n})$ using the baby-step giant-step algorithm [74] (see also [56, §3.1]).

For secrecy, each trustee $i \in n$ must demonstrate knowledge of a discrete logarithm $\log_g h_i$, that is, they prove that h_i has been correctly constructed; this prevents, for example, a trustee constructing their public key share $h_i = h$. For integrity of decryption, each trustee $i \in n$ must demonstrate equality between discrete logarithms $\log_g h_i$ and $\log_a k_i$; this prevents, for example, a trustee constructing the public key share $h_i = g^{m+x_i}$ and providing the partial decryption $k_i = a^{x_i}$. In addition, the voter must demonstrate that a valid vote

has been encrypted. These proofs can be achieved using signatures of knowledge (see Appendix A for details).

2.2 Protocol description

An election is created by naming an election officer, selecting a set of trustees, and generating a distributed public key pair. The election officer publishes, on the bulletin board, the public part of the trustees' key (and proof of correct construction), the candidate list $\tilde{t} = (t_1, \ldots, t_\ell) \cup \{\epsilon\}$ (where ϵ represents a vote of abstention), and the list of eligible voters $\widetilde{id} = (id_1, \ldots, id_n)$; the officer also publishes the *election fingerprint*, that is, the hash of these parameters. Informally, the steps that participants take during a run of Helios are as follows.

1. The voter launches a browser script that downloads the election parameters and recomputes the election fingerprint. The voter should verify that the fingerprint corresponds to the value published on the bulletin board. (This ensures that the script is using the trustees' public key; in particular, it helps prevent encrypting a vote with an adversary's public key. Such attacks have been discussed in the context of Direct Anonymous Attestation by Rudolph [67]; although, the vulnerability was discounted, in the trusted computing setting, by Leung, Chen & Mitchell [57].)

2. The voter inputs her vote $v \in \tilde{t}$ to the browser script, which creates a ballot consisting of her vote encrypted by the trustees' public key, and a proof that the ballot represents a permitted vote (this is needed because the ballots are never decrypted individually, in particular, it prevents multiple votes being encoded as a single ballot). The ballot is displayed to the voter.

3. The voter can audit the ballot to check if it really represents a vote for her chosen candidate; if she decides to do this, then the script provides her with the random data used in the ballot creation. She can then independently reconstruct her ballot and verify that it is indeed well-formed. The script provides some practical resistance against vote selling by refusing to cast audited ballots. See Benaloh [11, 12] for further details on ballot auditing.

4. When the voter has decided to cast her ballot, the script submits it to the election officer. The election officer authenticates the voter and checks that she is eligible to vote. The election officer also verifies the proof and publishes the ballot, appended with the voter's identity id, on the bulletin board. (In practice, the election officer also publishes the hash of the ballot, we omit this detail for brevity.)

5. Individual voters can check that their ballots appear on the bulletin board and, by verifying the proof, observers are assured that ballots represent permitted votes.

6. After some predefined deadline, the election officer homomorphically combines the ballots and publishes the encrypted tally on the bulletin board. Anyone can check that tallying is performed correctly.

Figure 1 Ballot construction by the browser script

Input: Cryptographic parameters (p, q, g), public key h, candidate list $\tilde{t} = (t_1, \ldots, t_\ell) \cup \{\epsilon\}$ and vote v.

Output: Encrypted vote $(a_1, b_1), \ldots, (a_\ell, b_\ell)$, signatures of knowledge $(\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1), \ldots, (\bar{a}_\ell, \bar{b}_\ell, \bar{c}_\ell, \bar{s}_\ell, \bar{a}'_\ell, \bar{b}'_\ell, \bar{c}'_\ell, \bar{s}'_\ell)$ and signature of knowledge $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$.

1. If $v \notin \tilde{t}$ then the script terminates.
2. Encode the vote v as a bitstring. For all $1 \leq i \leq \ell$, let
$$m_i = \begin{cases} 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases}$$
3. The bitstring representing the vote is encrypted. For all $1 \leq i \leq \ell$, let
$$(a_i, b_i) = (g^{r_i} \bmod p,\ g^{m_i} \cdot h^{r_i} \bmod p)$$
where $r_i \in_R \mathbb{Z}_q^*$.
4. For all $1 \leq i \leq \ell$, let $(\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_i, \bar{a}'_i, \bar{b}'_i, \bar{c}'_i, \bar{s}'_i)$ be a signature of knowledge demonstrating that the ciphertext (a_i, b_i) contains either 0 or 1, that is, each candidate can receive at most one vote.
5. Let $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$ be a signature of knowledge demonstrating that the ciphertext $(a_1 \cdot \ldots \cdot a_\ell,\ b_1 \cdot \ldots \cdot b_\ell)$ contains either 0 or 1, that is, at most one candidate receives one vote.

7. Each of the trustees publishes a partial decryption of the encrypted tally, together with a signature of knowledge proving the partial decryption's correct construction. Anyone can verify these proofs.
8. The election officer decrypts the tally and publishes the result. Anyone can check this decryption.

Formally, Step 2 is defined in Figure 1. (For simplicity the ballot construction algorithm in Figure 1 considers a vote $v \in \tilde{t}$, this can be generalised [7] to consider a vote $\tilde{v} \subseteq \tilde{t}$.) Checking voter eligibility (Step 4) is beyond the scope of Helios and Adida *et al.* [7] propose the use of existing infrastructure. The remaining steps follow immediately from the application of cryptographic primitives (see Section 2.1 for details).

2.3 Software implementation

Helios 3.0 is an extension of Helios 2.0 which adds numerous practical features, including: integration of authentication with various web-services (for example,

Facebook, GMail and Twitter), bulk voter registration using pre-existing electoral rolls, and simplification of administration with multiple trustees. Helios 3.0 has been implemented and is publicly available: `http://heliosvoting.org/`.

3 Attacking ballot secrecy

Ballot secrecy means a voter's vote is not revealed to anyone. We show that the Helios protocol does not satisfy this definition of ballot secrecy, by presenting an attack which allows an adversary to reveal a voter's vote. Moreover, we will show that formal definitions of ballot secrecy [49, 31, 9] are also violated.

Intuitively, an adversary may identify a voter's ballot on the bulletin board (using the voter's id) and recast this ballot by corrupting dishonest voters. The multiple occurrences of the voter's ballot will leak information in the tally and the adversary can exploit this knowledge to violate the voter's privacy. An informal description of the attack will now be presented in the case of three eligible voters and Section 3.3 considers a more realistic setting. (A formal analysis appears in Section 4.)

3.1 Attack description

Let us consider an election with candidates $t_1, \ldots, t_\ell$ and three eligible voters who have identities id_1, id_2 and id_3. Suppose that voters id_1 and id_2 are honest, and id_3 is a dishonest voter controlled by the adversary. Further assume that the honest voters have cast their ballots. The bulletin board entries are as follows:

$$\begin{array}{l} id_1, ciph_1, spk_1, spk'_1 \\ id_2, ciph_2, spk_2, spk'_2 \end{array}$$

where for $i \in \{1, 2\}$ we have

$$\begin{array}{lcl} ciph_i & = & (a_{i,1}, b_{i,1}), \ldots, (a_{i,\ell}, b_{i,\ell}) \\ spk_i & = & (\bar{a}_{i,1}, \bar{b}_{i,1}, \bar{c}_{i,1}, \bar{s}_{i,1}, \bar{a}'_{i,1}, \bar{b}'_{i,1}, \bar{c}'_{i,1}, \bar{s}'_{i,1}), \\ & & \quad \ldots, (\bar{a}_{i,\ell}, \bar{b}_{i,\ell}, \bar{c}_{i,\ell}, \bar{s}_{i,\ell}, \bar{a}'_{i,\ell}, \bar{b}'_{i,\ell}, \bar{c}'_{i,\ell}, \bar{s}'_{i,\ell}) \\ spk'_i & = & (\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_i, \bar{a}'_i, \bar{b}'_i, \bar{c}'_i, \bar{s}'_i) \end{array}$$

The value $ciph_i$ is the ith voter's encrypted vote, spk_i demonstrates that ciphertexts $(a_{i,1}, b_{i,1}), \ldots, (a_{i,\ell}, b_{i,\ell})$ contain either 0 or 1, and spk'_i demonstrates that $(a_{i,1} \cdot \ldots \cdot a_{i,\ell},\ b_{i,1} \cdot \ldots \cdot b_{i,\ell})$ contains either 0 or 1.

Exploiting the absence of ballot independence. The adversary observes the bulletin board and selects $ciph_k, spk_k, spk'_k$ such that id_k is the voter whose privacy will be compromised, where $k \in \{1, 2\}$. The adversary submits the ballot $ciph_k, spk_k, spk'_k$ and it immediately follows that the bulletin board is composed as follows:

$$id_1, ciph_1, spk_1, spk'_1$$
$$id_2, ciph_2, spk_2, spk'_2$$
$$id_3, ciph_k, spk_k, spk'_k$$

It is trivial to see that each bulletin board entry represents a permitted vote; that is, $spk_1, spk'_1, spk_2, spk'_2, spk_k, spk'_k$ all contain valid signatures of knowledge.

We have informally shown that Helios does not satisfy ballot independence (observing another voter's interaction with the election system allows a voter to cast the *same* vote), and this will now be exploited to violate privacy.

Violating privacy. The homomorphic addition of ballots reveals the encrypted tally $(a_{1,1} \cdot a_{2,1} \cdot a_{k,1},\ b_{1,1} \cdot b_{2,1} \cdot b_{k,1}), \ldots, (a_{1,\ell} \cdot a_{2,\ell} \cdot a_{k,\ell},\ b_{1,\ell} \cdot b_{2,\ell} \cdot b_{k,\ell})$ and, given the partial decryptions, these ciphertexts can be decrypted to reveal the number of votes for each candidate. Since there will be at least two votes for the candidate voter id_k voted for, the voter's vote can be revealed and hence privacy is not preserved. Moreover, the vote of the remaining honest voter will also be revealed.

A video demonstrating the attack against the Helios 3.0 implementation has been produced [76].

3.2 Variants exploiting malleability and key reuse

In the aforementioned attack description, the ballots cast by two voters are identical which may result in the detection of an attack. For a covert attack, the adversary may prefer to cast a distinct ballot. This can be achieved by exploiting the malleability of ballots. In particular, given a valid ballot

$$(a_1, b_1), \ldots, (a_\ell, b_\ell), (\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1), \ldots, (\bar{a}_\ell, \bar{b}_\ell, \bar{c}_\ell, \bar{s}_\ell, \bar{a}'_\ell, \bar{b}'_\ell, \bar{c}'_\ell, \bar{s}'_\ell), (\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}') \quad \text{(B1)}$$

the following ballots are also valid

$$(a_1, b_1), \ldots, (a_\ell, b_\ell), (\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1 + q, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1 + q), \ldots, (\bar{a}_\ell, \bar{b}_\ell, \bar{c}_\ell, \bar{s}_\ell + q, \bar{a}'_\ell, \bar{b}'_\ell, \bar{c}'_\ell, \bar{s}'_\ell + q), (\bar{a}, \bar{b}, \bar{c}, \bar{s} + q, \bar{a}', \bar{b}', \bar{c}', \bar{s}' + q) \quad \text{(B2)}$$

$$(a_{\pi(1)}, b_{\pi(1)}), \ldots, (a_{\pi(\ell)}, b_{\pi(\ell)}), (\bar{a}_{\pi(1)}, \bar{b}_{\pi(1)}, \bar{c}_{\pi(1)}, \bar{s}_{\pi(1)}, \bar{a}'_{\pi(1)}, \bar{b}'_{\pi(1)}, \bar{c}'_{\pi(1)}, \bar{s}'_{\pi(1)}), \ldots, (\bar{a}_{\pi(\ell)}, \bar{b}_{\pi(\ell)}, \bar{c}_{\pi(\ell)}, \bar{s}_{\pi(\ell)}, \bar{a}'_{\pi(\ell)}, \bar{b}'_{\pi(\ell)}, \bar{c}'_{\pi(\ell)}, \bar{s}'_{\pi(\ell)}), (\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}') \quad \text{(B3)}$$

where π is an arbitrary permutation over $\{1, \ldots, \ell\}$. Ballot B2 adds q to the response components of Ballot B1, this changes the ballot but not the vote. (It

is also possible to modify a subset of the response components.) However, this might be considered an implementation bug in Helios 3.0, rather than a theoretical issue, because the ballots are identical if considered as group elements. Replaying Ballot B3 has the advantage of casting a theoretical distinct ballot, since Ballot B3 represents a vote for a different candidate (with the exception of an abstention vote), and it is possible to compromise ballot secrecy in elections with three voters without abstention votes; however, more than one (modified) ballot may be required in elections with abstention votes. This variant of our attack also demonstrates a further violation of ballot independence in Helios: observing another voter's interaction with the election system allows a voter to cast a *different* vote (for example, a voter can cast a distinct vote from their boss). Both variants of the attack are particularly useful when the bulletin board includes the hash of the ballot (for example, in the Helios 3.0 implementation), rather than the complete ballot, because the hashes will be distinct.

An adversary may replay ballots in different elections, when the trustees' public key is reused and the candidate lists for each election are of equal length. This variant of the attack can be avoided if distinct keys are used for each election.

The variants of our attack in this section have all been successfully launched against the Helios 3.0 implementation.

3.3 Generalised attack and French election case study

Our attack demonstrates that the ballot of an arbitrary voter can be replayed by any other voter. In general, this does not reveal the voter's vote; but, some information is leaked, and colluding voters can replay sufficiently many ballots to leak the voter's vote. We will now discuss the feasibility of compromising ballot secrecy in a real-world election, focusing on the cost of an attack in French legislative elections, where each district elects a representative for the French National Assembly. Districts have several polling stations and each polling station individually announces its tally [40]; these tallies are published in local newspapers. The publication of tallies is typical of French elections at all levels; for example, from the election of mayor, to the presidential election.

In this (standard) voting configuration, an adversary can violate the ballot secrecy of a given voter by corrupting voters registered at the same polling station (for example, a coalition of neighbours or a family). The corrupted voters replay the ballot of the voter under attack, as previously explained. The motivation for restricting the selection of corrupted voters to the same polling station is twofold. Firstly, fewer corrupt voters are required to significantly influence the tally of an individual polling station (in comparison to influencing the election outcome). Secondly, it is unlikely to change the district's elected representative, because a candidate will receive only a few additional votes in the district; it follows that coercing voters to sacrifice their vote, for the purposes of the attack, should be easier. In the remainder of this section, we discuss how many corrupt voters are required to violate ballot secrecy – by making a significant change in the tally of a polling station – in an arbitrary district of

Party	Tally
PS	4120
UMP	3463
FN	1933
Europe Eco.	1921
Front de gauche	880
NPA	697
MODEM	456
Debout la République	431
Alliance école	193
LO	156
Émergence	113
Liste chrétienne	113

Table 1: 2010 legislative election results in Aulnay-sous-Bois [41]

Aulnay-sous-Bois and a rural district in Toul.

3.3.1 Ballot secrecy in Aulnay-sous-Bois

Using historic data and/or polls, it is possible to construct the expected distribution of votes. For simplicity, let us assume the distribution of votes per polling station is the average of the 2010 tally (Table 1), and that if the adversary can increase the number of votes for a particular candidate by more than σ (by replaying a voter's ballot), then this is sufficient to determine that the voter voted for that candidate. In addition, suppose that the adversary corrupts abstaining voters and therefore we do not consider the redistribution of votes. We remark that corrupting abstaining voters may be a fruitful strategy, since abstaining voters do not sacrifice their vote by participating in an attack.

Table 2 presents the expected distribution of votes, and includes the number of voters that an adversary must corrupt to determine if a voter voted for a particular candidate, for various values of σ. We shall further assume that participation in the region is consistent with 2010; that is, 291 of the 832 eligible voters are expected to participate. It follows that 50 voters corresponds to approximately 6% of the Aulnay-sous-Bois electorate, and 10 voters corresponds to approximately 1%. Our results therefore demonstrate that the privacy of a voter can be compromised by corrupting a small number of voters. In particular, for medium-size parties (in terms of votes received) – including, for example, FN and Europe Ecologie – it is sufficient to corrupt 19 voters to see the number of votes increase by 50%. Furthermore, given the low turn-out (541 voters are expected to abstain), it seems feasible to corrupt abstaining voters, and therefore an attack can be launched without any voter sacrificing their vote.

Party	Expected tally	$\sigma = 200\%$	$\sigma = 150\%$	$\sigma = 50\%$	$\sigma = 20\%$
PS	81	162	122	41	17
UMP	68	136	102	34	14
FN	38	76	57	19	8
Europe Eco.	38	76	57	19	8
Front de gauche	17	34	26	9	4
NPA	14	28	21	7	3
MODEM	9	18	14	5	2
Debout la République	8	16	12	4	2
Alliance école	4	8	6	2	1
LO	3	6	5	2	1
Émergence	2	4	3	1	1
Liste chrétienne	2	4	3	1	1

Table 2: Number of duplicate ballots for a significant change in the tally

Limitations. For such an attack based on a statistical model, we acknowledge that this model is rather naïve, but believe it is sufficiently indicative to illustrate the real threat of an attack against privacy. A definitive mathematical analysis should be considered in the future.

Cases of complete privacy breach. The probabilistic nature of these attacks may introduce sufficient uncertainty to prevent privacy violations, and we will consider voting configurations where an adversary can definitively learn a voter's vote. Observe that if an attacker can corrupt half of the voters at a polling station, then the vote of an arbitrary voter can be revealed. Moreover, the cost of this attack can be reduced. In particular, if n dishonest voter's replay voter $\mathcal{V}$'s ballot, then it is possible to deduce that $\mathcal{V}$ did not vote for any candidate that received strictly less than $n+1$ votes. This leaks information about voter $\mathcal{V}$'s chosen candidate and in cases where exactly one candidate received more than n votes, the voter's vote can be deduced.

3.3.2 Ballot secrecy in small polling stations

The difficulties of large scale corruption may prohibit our attack in the majority of polling stations; however, our attack is feasible in small polling stations found in rural districts. For example, let us consider the 2007 legislative elections in the district of Toul [38]. This district has 75350 eligible voters registered at 193 polling stations. Accordingly, the average polling station has 390 registered voters, but the variance is large. Indeed, 33 polling stations have between 50 and 99 voters, 9 polling stations have less then 50 voters, and the smallest two polling stations have 8, respectively 16, voters. Moreover, the attack is simplified by non-participating voters. In these small polling stations it is thus sufficient to corrupt a very small number of voters to reveal a voter's vote while the final outcome of the election would not change as it is based on 75350 eligible voters.

4 Solution: Weeding replayed ballots

Our attack exploits the possibility of replaying a voter's ballot without detection, and can be attributed to the lack of ballot independence in Helios. This section sketches some possible solutions to ensure ballot independence.

4.1 Weeding replayed ballots

The ballots replayed in our attacks can all be identified. First, ciphertexts and signatures of knowledge should have a unique representation as group elements, for example, by requiring that the response component of signatures of knowledge is in the interval $[0, q-1]$. Second, a ballot should not contain a ciphertext that already exists on the bulletin board. The election officer should reject ballots that do not satisfy these conditions. This solution is simple and can easily be implemented in a future version of Helios.

4.2 Binding ballots to voters

The previous approach requires a special mechanism to handle replayed ballots. We now propose a technique that makes such actions futile. In essence, based upon inspiration from [43, §4.2] and [29], we ensure that proofs associated with replayed ballots are considered invalid; that is, we bind the link between a voter and her ballot. This is achieved by adding the identity of the voter in the construction of challenges used by signatures of knowledge. More precisely, for voter id, the sign algorithm (defined in Appendix A) is modified as follows: on input (a, b), such that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$, let challenge $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) - \sum_{i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}} c_i \pmod q$, where values $a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}$ and $c_1, \ldots, c_{m-1}, c_{m+1}, \ldots, c_m$ are defined as before. For correctness, the verification algorithm must also be modified. In particular, for candidate signatures constructed by voter id, the verifier should check $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) \equiv \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \pmod q$.

In a similar direction, the electronic voting protocol proposed by Juels, Catalano & Jakobsson [47] – which has been implemented by Clarkson, Chong & Myers [24, 23] as Civitas – requires ballots to be bound to private voter credentials. This provides *eligibility verifiability* [50]: anyone can check that each ballot published on the bulletin board was cast by a registered voter and at most one ballot is tallied per voter. It is likely that eligibility verifiability enforces ballot independence, but the provision of eligibility verifiability appears to be expensive, in particular, Juels, Catalano & Jakobsson and Clarkson, Chong & Myers assume the existence of an infrastructure for voter credentials.

4.3 Discussion

Our *weeding replayed ballots* solution is particular attractive because it adheres to Benaloh's notion of *ballot casting assurance* [11, 12] which asserts that the ballot encryption device (the browser script in this instance) does not know the voter's identity. (We remark that neither the original Helios scheme nor our proposed fix strictly satisfy Benaloh's notion of ballot casting assurance if a voter decides to use her own computer.) The ballot casting assurance principle is important because knowledge of the voter's identity could be used to infer the likelihood of auditing and this information can be used to influence the behaviour of the ballot encryption device; in particular, if a ballot is unlikely to be audited, then the device may act maliciously, for example, by encrypting a different vote. By comparison, the *binding ballots to voters* solution would necessarily require that the voter's identity is revealed to the ballot encryption device. Moreover, for privacy purposes, the election officer may chose to allocate voters with pseudo-identities when casting ballots (rather than associate ballots with actual voter identities); since these pseudo-identities are unknown to voters in advance, an additional interaction with the election officer would be required. (Note that the use of pseudo-identities does not prevent the attack by breaking the link between ballots and voters, because the link is known by the election officer.) Finally, extending Helios to provide eligibility verifiability

would require a considerable extension to the Helios code-base and, furthermore, finding a suitable solution is an open problem. Accordingly, we adopt the weeding replayed ballot solution and, in the next section, we show that this is sufficient to ensure ballot secrecy, in the formal setting.

5 Formal proof of ballot secrecy

We formally prove that weeding duplicate ballots ensures ballot secrecy. We make use of the applied pi calculus [1, 68], due to its proven suitability for evaluating security properties of electronic voting protocols (see, for example, [31, 9, 50]).

5.1 Applied pi calculus

We first recall the applied pi calculus setting [1]. We assume an infinite set of *names* $a, b, c, \ldots, k, \ldots, m, n, \ldots, s, \ldots$, an infinite set of *variables* $x, y, z, \ldots$, and a *signature* Σ consisting of a finite set of *function symbols*, each with an associated arity. We use metavariables u, w to range over both names and variables. *Terms* L, M, N, T, U, V are built by applying function symbols to names, variables, and other terms. We write $\{^M/_x\}$ for the *substitution* that replaces the variable x with the term M. Arbitrarily large substitutions can be written as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ and the letters σ and τ range over substitutions. We write $N\sigma$ for the result of applying σ to the free variables of term N. A term is *ground* when it does not contain variables.

The signature Σ is equipped with an *equational theory* E, that is, a set of equations of the form $M = N$, where the terms M, N are defined over the signature Σ. We define equality modulo the equational theory, written $=_E$, as the smallest equivalence relation on terms that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names. We write $M =_E N$ when the equation $M = N$ is in the theory E, and keep the signature implicit. When E is clear from its usage, we may abbreviate $M =_E N$ as $M = N$. The negation of $M =_E N$ is denoted $M \neq_E N$ (and similarly abbreviated $M \neq N$).

Processes and *extended processes* are defined in the usual way (Figure 2). We write $\nu\, \tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu\, u_1. \cdots .\nu\, u_l$. The active substitution $\{^M/_x\}$ can replace the variable x for the term M in every process it comes into contact with and this behaviour can be controlled by restriction, in particular, the process $\nu\, x.(\{^M/_x\} \mid P)$ corresponds exactly to let $x = M$ in P. Arbitrarily large active substitutions can be obtained by parallel composition and we occasionally abbreviate $\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_l}/_{x_l}\}$ as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ or $\{^{\tilde{M}}/_{\tilde{x}}\}$. We also use σ and τ to range over active substitutions, and write $N\sigma$ for the result of applying σ to the free variables of N. Extended processes must have at most one active substitution for each variable and there is exactly one when the variable is under restriction. The only minor change compared to [1] is that conditional branches now depend on formulae

Figure 2 Syntax for processes

$P, Q, R ::=$	(plain) processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu\, n.P$	name restriction
if ϕ then P else Q	conditional
$u(x).P$	message input
$\overline{u}\langle M\rangle.P$	message output
$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu\, n.A$	name restriction
$\nu\, x.A$	variable restriction
$\{^M/_x\}$	active substitution

$\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$. If M and N are ground, we define $[\![M = N]\!]$ to be true if $M =_E N$ and false otherwise. The semantics of $[\![\ ]\!]$ is then extended to formulae in the standard way.

The *scope* of names and variables are delimited by binders $u(x)$ and $\nu\, u$. The set of bound names is written $\mathrm{bn}(A)$ and the set of bound variables is written $\mathrm{bv}(A)$; similarly we define the set of free names $\mathrm{fn}(A)$ and free variables $\mathrm{fv}(A)$. Occasionally, we write $\mathrm{fn}(M)$ (and $\mathrm{fv}(M)$ respectively) for the set of names (and respectively variables) which appear in term M. An extended process is *closed* when every variable x is either bound or defined by an active substitution.

We define a *context* $C[\_]$ to be an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with the extended process A. An *evaluation context* is a context whose hole is not in the scope of a replication, a conditional, an input, or an output. A context $C[\_]$ closes A when $C[A]$ is closed.

A *frame*, denoted φ or ψ, is an extended process built from the null process 0 and active substitutions $\{^M/_x\}$, which are composed by parallel composition and restriction. The *domain* $\mathrm{dom}(\varphi)$ of a frame φ is the set of variables that φ exports, that is, the set of variables x for which φ contains an active substitution $\{^M/_x\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

5.1.1 Operational semantics

The operational semantics are defined by three relations: *structural equivalence* ($\equiv$), *internal reduction* ($\rightarrow$), and *labelled reduction* ($\xrightarrow{\alpha}$). These relations satisfy the rules in Figure 3 and are defined such that: structural equivalence is the smallest equivalence relation on extended processes that is closed by α-

Figure 3 Semantics for processes

$$
\begin{array}{lrcl}
\textsc{Par-0} & A & \equiv & A \mid 0 \\
\textsc{Par-A} & A \mid (B \mid C) & \equiv & (A \mid B) \mid C \\
\textsc{Par-C} & A \mid B & \equiv & B \mid A \\
\textsc{Repl} & !P & \equiv & P \mid !P
\end{array}
$$

$$
\begin{array}{lrcl}
\textsc{New-0} & \nu\, n.0 & \equiv & 0 \\
\textsc{New-C} & \nu\, u.\nu\, w.A & \equiv & \nu\, w.\nu\, u.A \\
\textsc{New-Par} & A \mid \nu\, u.B & \equiv & \nu\, u.(A \mid B) \\
 & & & \text{where } u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)
\end{array}
$$

$$
\begin{array}{lrcl}
\textsc{Alias} & \nu\, x.\{^M/_x\} & \equiv & 0 \\
\textsc{Subst} & \{^M/_x\} \mid A & \equiv & \{^M/_x\} \mid A\{^M/_x\} \\
\textsc{Rewrite} & \{^M/_x\} & \equiv & \{^N/_x\} \\
 & & & \text{where } M =_E N
\end{array}
$$

$$
\begin{array}{ll}
\textsc{Comm} & \overline{c}\langle x\rangle.P \mid c(x).Q \to P \mid Q \\
\textsc{Then} & \text{if } \phi \text{ then } P \text{ else } Q \to P \quad \text{if } [\![\phi]\!] = \mathsf{true} \\
\textsc{Else} & \text{if } \phi \text{ then } P \text{ else } Q \to Q \quad \text{otherwise}
\end{array}
$$

$$\textsc{In} \quad c(x).P \xrightarrow{c(M)} P\{^M/_x\}$$

$$\textsc{Out-Atom} \quad \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P$$

$$\textsc{Open-Atom} \quad \frac{A \xrightarrow{\overline{c}\langle u\rangle} A' \qquad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'}$$

$$\textsc{Scope} \quad \frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'}$$

$$\textsc{Par} \quad \frac{A \xrightarrow{\alpha} A' \qquad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\textsc{Struct} \quad \frac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

conversion of both bound names and bound variables, and closed under application of evaluation contexts; internal reduction is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts; and for labelled reductions α is a *label* of the form $c(M)$, $\overline{c}\langle u\rangle$, or $\nu\, u.\overline{c}\langle u\rangle$ such that u is either a channel name or a variable of base type.

5.1.2 Equivalence

The definition of observational equivalence [1] quantifies over all contexts which makes proofs difficult, therefore we adopt labelled bisimilarity in this article. Labelled bisimilarity relies on an equivalence relation between frames, called static equivalence.

Definition 1 (Static equivalence)**.** *Two closed frames φ and ψ are statically equivalent, denoted $\varphi \approx_s \psi$, if $\mathrm{dom}(\varphi) = \mathrm{dom}(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions σ, τ such that $\varphi \equiv \nu\, \tilde{n}.\sigma$ and $\psi \equiv \nu\, \tilde{n}.\tau$ and for all terms M, N such that $\tilde{n} \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. Two closed extended processes A, B are statically equivalent, written $A \approx_s B$, if their frames are statically equivalent; that is, $\varphi(A) \approx_s \varphi(B)$.*

The relation $\approx_s$ is called *static* equivalence because it only examines the current state of the processes, and not the processes' dynamic behaviour. The following definition of labelled bisimilarity captures the dynamic part.

Definition 2 (Labelled bisimilarity)**.** *Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\, \mathcal{R}\, B$ implies:*

1. *$A \approx_s B$;*
2. *if $A \to A'$, then $B \to^* B'$ and $A'\, \mathcal{R}\, B'$ for some B';*
3. *if $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A'\, \mathcal{R}\, B'$ for some B'.*

Definitions of observational equivalence and labelled bisimilarity have been shown to coincide [58].

5.2 Modelling Helios in applied pi

We start by constructing a suitable signature Σ to capture the cryptographic primitives used by Helios and define an equational theory E to capture the relationship between these primitives.

5.2.1 Signature

We adopt the following signature.

$$\Sigma = \{\mathsf{ok}, \mathsf{zero}, \mathsf{one}, \bot, \mathsf{fst}, \mathsf{snd}, \mathsf{pair}, *, +, \circ, \mathsf{partial}, \mathsf{checkspk}, \mathsf{penc}, \mathsf{spk}, \mathsf{dec}\}$$

Functions ok, zero, one, $\bot$ are constants; fst, snd are unary functions; dec, pair, partial, $*$, $+$, $\circ$ are binary functions; checkspk, penc are ternary functions; and spk is a function of arity four. We adopt infix notation for $*$, $+$, and $\circ$.

The term $\mathsf{penc}(T, N, M)$ denotes the encryption of plaintext M, using random nonce N and key T. The term $U * U'$ denotes the homomorphic combination of ciphertexts U and U', the corresponding operation on plaintexts is written

$M + M'$ and $N \circ N'$ on nonces. The partial decryption of ciphertext U using key L is denoted $\mathsf{partial}(L, U)$. The term $\mathsf{spk}(T, N, M, U)$ represents a signature of knowledge that proves U is a ciphertext under the public key T on the plaintext M using nonce N and such that M is either the constant zero or one. We introduce tuples using pairings and, for convenience, we occasionally abbreviated $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_n, \bot)))$ as $(\!|M_1, \ldots, M_n|\!)$, and $\mathsf{fst}(\mathsf{snd}^{i-1}(M))$ is denoted $\pi_i(M)$, where $i \in \mathbb{N}$. We use the equational theory E that asserts functions $+, *, \circ$ are commutative and associative, and includes the equations:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \tag{E1}$$

$$\mathsf{snd}(\mathsf{pair}(x, y)) = y \tag{E2}$$

$$\mathsf{zero} + \mathsf{one} = \mathsf{one} \tag{E3}$$

$$\mathsf{zero} + \mathsf{zero} = \mathsf{zero} \tag{E4}$$

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})) = x_{\mathsf{plain}} \tag{E5}$$

$$\mathsf{dec}(\mathsf{partial}(x_{\mathsf{sk}}, \mathit{ciph}), \mathit{ciph}) = x_{\mathsf{plain}} \quad \text{where } \mathit{ciph} = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}}) \tag{E6}$$

$$\mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}}, y_{\mathsf{plain}}) * \mathsf{penc}(x_{\mathsf{pk}}, z_{\mathsf{rand}}, z_{\mathsf{plain}}) = \mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}} \circ z_{\mathsf{rand}}, y_{\mathsf{plain}} + z_{\mathsf{plain}}) \tag{E7}$$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero}, \mathit{ball})) = \mathsf{ok} \quad \text{where } \mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero}) \tag{E8}$$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one}, \mathit{ball})) = \mathsf{ok} \quad \text{where } \mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one}) \tag{E9}$$

Equation E6 allows plaintext M to be recovered from ciphertext $\mathsf{penc}(\mathsf{pk}(L), N, M)$ given partial decryption $\mathsf{partial}(L, \mathsf{penc}(\mathsf{pk}(L), N, M))$, when the partial decryption is constructed using the private key L. Equation E7 represents the homomorphic combination of ciphertexts. The Equations E8 and E9 allow the verification of signatures of knowledge $\mathsf{spk}(T, N, M, \mathsf{penc}(T, N, M))$, when $M \in \{\mathsf{zero}, \mathsf{one}\}$. The remaining equations are standard.

Example 1. *Given randomness N, N', plaintexts $(M, M') \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$, and public key T, one can construct a signature of knowledge $L = \mathsf{spk}(T, N \circ N', M + M', \mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M'))$. Then* checkspk *applied to the public key T, the homomorphically combined ciphertexts $\mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M')$, and the signature L is equal to* ok *using Equations E3, E7, E8, and E9*

5.2.2 Helios process specification

In the applied pi calculus, it is sufficient to model the parts of the voting system which need to be trusted for ballot secrecy; all the remaining parts of the system are controlled by the adversarial environment. Accordingly, we assume the

existence of at least two honest voters $\mathcal{A}$, $\mathcal{B}$; since this avoids the scenario where ballot secrecy of an individual voter is compromised by collusion amongst all the remaining voters. In addition, the following trust assumptions are required.

- At least one trustee is honest
- The election officer runs the bulletin board honestly:
  - Voters $\mathcal{A}$, $\mathcal{B}$ have authentic channels with the bulletin board
  - Signatures of knowledge are checked for dishonest voters*
  - Replays of honest ballots (that is, those cast by $\mathcal{A}$ or $\mathcal{B}$) are rejected*
  - The tally is correctly computed*
  - The trustees have an authentic channel with the bulletin board
- The browser script is trusted and has the correct public key of the election

(Assumptions marked with * could be performed by an honest trustee, rather than the bulletin board.) Although neither voters nor observers can verify that there exists an honest trustee, an assurance of trust is provided by distribution. The necessity to trust the election officer to run the bulletin board is less desirable and work-in-progress [64] aims to weaken this assumption; moreover, to further distribute trust assumptions, the trustees could also check signatures and tallying. Finally, trust in the browser script can be obtained by using software written by a reputable source or writing your own code.

The trusted components are modelled by the administration process $A^{\phi}_{\ell,n}$ and voting process V_ℓ defined in Figure 4. For generality, the voting process V_ℓ is parametrised by the number of candidates ℓ. Similarly, the administration process $A^{\phi}_{\ell,n}$ is parametrised by the number of candidates ℓ, the number of voters n, and a formula ϕ; the formula ϕ defines the checks performed by the bulletin board before accepting a ballot. We will consider several variants of Helios (including the original Helios 2.0 protocol and our fixed scheme) by considering suitable formula that we call *Helios process specifications*.

Definition 3 (Helios process specification)**.** *A formula $\phi_{\ell,\bar{n}}$ is a* Helios process specification*, if* $\mathrm{fv}(\phi_{\ell,\bar{n}}) \subseteq \{y_1, \ldots, y_{\bar{n}}, y_{\mathsf{ballot}}, z_{\mathsf{pk}}\}$.

The voting process V_ℓ contains free variables $x_1^{\mathsf{vote}}, \ldots, x_\ell^{\mathsf{vote}}$ to represent the voter's vote (which is expected to be encoded using constants zero and one) and the free variable x_{auth} represents the channel shared by the voter and the bulletin board. The definition of the process V_ℓ corresponds to the description of the browser script (Figure 1). The administration process $A^{\phi}_{\ell,n}$ is parametrised by the number of candidates ℓ, the number of voters n, and a Helios process specification ϕ. The restricted name sk_T models the tallier's secret key and the public part $\mathsf{pk}(sk_T)$ is included in the process's frame. The restricted names a_1 and a_2 model authentic channels between the two honest voters and the bulletin board, and the channel name d captures the authentic channel with the honest

Figure 4 Helios process specification

Let ℓ be some number of candidates, $n \geq 2$ be some number of voters, and ϕ be a Helios process specification. The administration process $A^{\phi}_{\ell,n}$ and voting process V_ℓ are defined below.

$$
\begin{array}{lcl}
V_\ell & = & \nu\, r_1\ . \\
 & & \mathrm{let}\ ciph_1 = \mathsf{penc}(z_{\mathsf{pk}}, r_1, x_1^{\mathsf{vote}})\ \mathrm{in} \\
 & & \mathrm{let}\ spk_1 = \mathsf{spk}(z_{\mathsf{pk}}, r_1, x_1^{\mathsf{vote}}, ciph_1))\ \mathrm{in} \\
 & & \vdots \\
 & & \nu\, r_\ell\ . \\
 & & \mathrm{let}\ ciph_\ell = \mathsf{penc}(z_{\mathsf{pk}}, r_\ell, x_\ell^{\mathsf{vote}})\ \mathrm{in} \\
 & & \mathrm{let}\ spk_\ell = \mathsf{spk}(z_{\mathsf{pk}}, r_\ell, x_\ell^{\mathsf{vote}}, ciph_\ell))\ \mathrm{in} \\
 & & \mathrm{let}\ \hat{r} = r_1 \circ \cdots \circ r_\ell\ \mathrm{in} \\
 & & \mathrm{let}\ \widehat{ciph} = ciph_1 * \cdots * ciph_\ell\ \mathrm{in} \\
 & & \mathrm{let}\ \widehat{vote} = x_1^{\mathsf{vote}} + \cdots + x_\ell^{\mathsf{vote}}\ \mathrm{in} \\
 & & \mathrm{let}\ \widehat{spk} = \mathsf{spk}(z_{\mathsf{pk}}, \hat{r}, \widehat{vote}, \widehat{ciph})\ \mathrm{in} \\
 & & \overline{x_{\mathsf{auth}}}\langle (\!| ciph_1, \ldots, ciph_\ell,\ spk_1, \ldots, spk_\ell,\ \widehat{spk} |\!) \rangle \\
 & & \\
A^{\phi}_{\ell,n} & = & \nu\, sk_T, a_1, a_2, d\ .\ (\_ \mid BB^{\phi}_{\ell,n} \mid T_\ell \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}) \\
 & & \\
BB^{\phi}_{\ell,n} & = & a_1(y_1)\ .\ \overline{c}\langle y_1 \rangle\ .\ a_2(y_2)\ .\ \overline{c}\langle y_2 \rangle\ . \\
 & & a_3(y_3)\ .\ \mathrm{if}\ \phi_{\ell,2}\{^{y_3}/_{y_{\mathsf{ballot}}}\}\ \mathrm{then} \\
 & & \cdots\ a_n(y_n)\ .\ \mathrm{if}\ \phi_{\ell,n-1}\{^{y_n}/_{y_{\mathsf{ballot}}}\}\ \mathrm{then} \\
 & & \mathrm{let}\ tally_1 = \pi_1(y_1) * \cdots * \pi_1(y_n)\ \mathrm{in} \\
 & & \cdots \mathrm{let}\ tally_\ell = \pi_\ell(y_1) * \cdots * \pi_\ell(y_n)\ \mathrm{in} \\
 & & \overline{d}\langle (\!| tally_1, \ldots, tally_\ell |\!) \rangle\ . \\
 & & d(y_{\mathsf{partial}})\ . \\
 & & \overline{c}\langle y_{\mathsf{partial}} \rangle\ . \\
 & & \overline{c}\langle (\!| \mathsf{dec}(\pi_1(y_{\mathsf{partial}}), tally), \ldots, \mathsf{dec}(\pi_\ell(y_{\mathsf{partial}}), tally') |\!) \rangle \\
 & & \\
T_\ell & = & d(y_{\mathsf{tally}})\ . \\
 & & \overline{d}\langle (\!| \mathsf{partial}(sk_T, \pi_1(y_{\mathsf{tally}})), \ldots, \mathsf{partial}(sk_T, \pi_\ell(y_{\mathsf{tally}})) |\!) \rangle
\end{array}
$$

trustee. To ensure the adversary has access to messages sent on private channels, communication is relayed on the public channel c. The sub-process $BB^{\phi}_{\ell,n}$ represents the bulletin board and T_ℓ represents the tallier. The bulletin board accepts ballots from each voter and checks they are valid using the Helios process specification ϕ (this predicate will be discussed in more detail below). Once all ballots have been submitted, the bulletin board homomorphically combines the ciphertexts and sends the encrypted tallies to the tallier for decryption. (The necessity for all voters to participate is included for simplicity, in particular, our bulletin board does not weed ballots containing invalid proofs.) The tallier

receives the homomorphic combinations of ballots y_{tally} and derives a partial decryption for each candidate; these partial decryptions are sent to the bulletin board and the election result is published.

The voting process V_ℓ is parameterised by a substitution σ, where variables $x_1^{\mathsf{vote}}, \ldots, x_\ell^{\mathsf{vote}} \in \mathrm{dom}(\sigma)$; these variables must be parameterised to encode a vote for at most one candidate, that is, there exists at most one integer $i \in \{1, \ldots, \ell\}$ such that $\Sigma \vdash x_i^{\mathsf{vote}}\sigma = \mathsf{one}$. Formally, we define valid parameterisations using the notion of *candidate substitutions*.

Definition 4 (Candidate substitution)**.** *Given some number of candidates ℓ and a substitution σ, we say σ is a* candidate substitution *if*

$$\Sigma \vdash (x_1^{\mathsf{vote}} + \cdots + x_\ell^{\mathsf{vote}})\sigma = \mathsf{zero} \vee \Sigma \vdash (x_1^{\mathsf{vote}} + \cdots + x_\ell^{\mathsf{vote}})\sigma = \mathsf{one}$$

It follows immediately that bitstrings $m_1, \ldots, m_\ell$ generated during Step 2 of Figure 1 can be modelled as candidate substitutions.

The application of our model is demonstrated in the following example.

Example 2. *Let ℓ be some number of candidates, $n \geq 2$ be some number of voters, and ϕ be a Helios process specification. An election with voters $\mathcal{A}$ and $\mathcal{B}$ who select candidate substitutions σ and τ, and such that the other $n-2$ voters are controlled by the adversary, can be modelled by the process $A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$.*

Ballot validity. In Helios 2.0, the election officer considers a ballot to be valid if the signature proofs of knowledge hold. Accordingly, we can model the Helios administration by the process $A^{\phi^{\mathsf{orig}}}_{\ell,n}$ where the Helios process specification ϕ^{orig}, parameterised by the number of candidates ℓ, is defined as follows.

$$\begin{aligned}\phi_\ell^{\mathsf{orig}} \triangleq\ & \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}) * \cdots * \pi_\ell(y_{\mathsf{ballot}}), \pi_{2\cdot\ell+1}(y_{\mathsf{ballot}})) = \mathsf{ok} \wedge \\ & \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}), \pi_{\ell+1}(y_{\mathsf{ballot}})) = \mathsf{ok} \wedge \ldots \wedge \\ & \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_\ell(y_{\mathsf{ballot}}), \pi_{2\cdot\ell}(y_{\mathsf{ballot}})) = \mathsf{ok}\end{aligned}$$

We have shown that these checks are insufficient to ensure ballot secrecy (Section 3). Our weeding replayed ballots solution, proposed in Section 4.1, additionally requires that the ciphertexts inside the ballot do not appear on the bulletin board. This revised scheme can be modelled using the Helios process specification ϕ^{sol}, parameterised by the number of candidates ℓ and number of ballots already on the bulletin board $\bar{n}$, defined as follows.

$$\phi^{\mathsf{sol}}_{\ell,\bar{n}} \triangleq \phi^{\mathsf{orig}}_\ell \wedge \pi_{2\cdot\ell+2}(y_{\mathsf{ballot}}) = \bot \wedge \bigwedge_{\substack{i,j\in\{1,\ldots,\ell\},\\ k\in\{1,\ldots,\bar{n}\}}} \pi_i(y_k) \neq \pi_j(y_{\mathsf{ballot}})$$

We can also model a naïve solution that would consist in weeding only identical ballots by considering the Helios process specification ϕ^{ident}, parameterised by

the number of candidates ℓ and number of ballots already on the bulletin board $\bar{n}$, defined below.

$$\phi^{\mathsf{ident}}_{\ell,\bar{n}} \triangleq \phi^{\mathsf{orig}}_{\ell} \wedge \pi_6(y_{\mathsf{ballot}}) = \bot \wedge \bigwedge_{k \in \{1,\ldots,\bar{n}\}} y_{\mathsf{ballot}} \neq y_k$$

We have already shown that removing exact duplicates is insufficient because it would fail to detect variants of our attack whereby the contents of a ballot are permuted. In the next section, we formally show that Helios 2.0 (modelled using ϕ^{orig}) and the naïve solution (modelled using ϕ^{ident}) do not satisfy ballot secrecy, and that our proposed solution (modelled using ϕ^{sol}) does satisfy ballot secrecy.

5.3 Formal analysis: Ballot secrecy

Based upon [49, 30, 31], and as previous discussed (see *related work* in Section 1), we formalise ballot secrecy for two voters $\mathcal{A}$ and $\mathcal{B}$ with the assertion that an adversary cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. Formally, this is captured by Definition 5.

Definition 5 (Ballot secrecy)**.** *Given a Helios process specification ϕ, we say* ballot secrecy is satisfied *if for all integers $\ell \in \mathbb{N}^*$ and $n \geq 2$, and for all candidates substitutions σ and τ, we have*

$$A^{\phi}_{\ell,n}[V_\ell\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V_\ell\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\tau] \approx_l A^{\phi}_{\ell,n}[V_\ell\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V_\ell\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$$

The ballot secrecy definition proposed by Delaune, Kremer & Ryan considered a vote to be an arbitrary name, whereas a vote in our setting must be a series of the constant symbols zero and one, such that their combination by application of the function $+$ is also a constant zero and one; it follows that Definition 5 is a straightforward variant of the original.

The Helios 2.0 protocol does not satisfy our privacy definition (Lemma 1) and naïve ballot weeding solutions are also insufficient (Lemma 2).

Lemma 1. *The Helios process specification ϕ^{orig} does not satisfy ballot secrecy.*

Intuitively, the proof of Lemma 1 is due to the environment's ability to replay $\mathcal{A}$'s ballot, therefore introducing an observable difference: the result will include two instances of $\mathcal{A}$'s vote. Formally, this follows immediately from the proof Lemma 2.

Lemma 2. *The Helios process specification ϕ^{ident} does not satisfy ballot secrecy.*

Proof. Consider $\ell = 2$, $n = 3$, $\sigma = \{{}^{\mathsf{zero}}/_{x_1^{\mathsf{vote}}}, {}^{\mathsf{one}}/_{x_2^{\mathsf{vote}}}\}$ and $\tau = \{{}^{\mathsf{one}}/_{x_1^{\mathsf{vote}}}, {}^{\mathsf{zero}}/_{x_2^{\mathsf{vote}}}\}$. We consider a sequence of transitions where the two voters output their ballots and then the adversary chooses its ballots to be a permutation of the first voter's ballot. Namely, if the first voter's ballot is $(\!|ciph, ciph', spk, spk', \widehat{spk}|\!)$ then the adversary outputs $(\!|ciph', ciph, spk', spk, \widehat{spk}|\!)$. Formally, this corresponds to the following transitions

$$A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$$
$$\to \xrightarrow{\nu\, x.\overline{c}\langle x\rangle} \to \xrightarrow{\nu\, y.\overline{c}\langle y\rangle} \xrightarrow{c(\langle\!|\pi_2(x),\pi_1(x),\pi_4(x),\pi_3(x),\pi_5(x)|\!\rangle)} \to^* \xrightarrow{\nu\, z.\overline{c}\langle z\rangle} \nu\,\tilde{n}.\tau_1$$

for some names $\tilde{n}$ and substitution τ_1, such that:

$$\mathsf{dec}(\pi_1(z), \pi_1(x) * \pi_1(y) * \pi_2(x))\tau_1 =_E \mathsf{one} + \mathsf{one}$$

This labeled transition has to matched by

$$A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$$
$$\to \xrightarrow{\nu\, x.\overline{c}\langle x\rangle} \to \xrightarrow{\nu\, y.\overline{c}\langle y\rangle} \xrightarrow{c(\langle\!|\pi_2(x),\pi_1(x),\pi_4(x),\pi_3(x),\pi_5(x)|\!\rangle)} \to^* \xrightarrow{\nu\, z.\overline{c}\langle z\rangle} \nu\,\tilde{n}.\tau_2$$

for some names $\tilde{n}$ and substitution τ_2, such that:

$$\mathsf{dec}(\pi_1(z), \pi_1(x) * \pi_1(y) * \pi_2(x))\tau_2 =_E \mathsf{one}$$

It follows immediately that $\nu\,\tilde{n}.\tau_1 \not\approx_s \nu\,\tilde{n}.\tau_2$ and, hence, ϕ^{ident} does not satisfy ballot secrecy. □

In contrast, removing duplicates up to permutation ensures ballot secrecy.

Theorem 1. *The Helios process specification ϕ^{sol} satisfies ballot secrecy.*

ProVerif is an automatic tool that can check equivalence in the applied pi calculus [16]. Although ProVerif has been successfully used to prove ballot secrecy (for example, in the Fujioka, Okamoto & Ohta protocol [33]), it cannot prove Theorem 1, at the time of writing, for two main reasons. Firstly, ProVerif cannot prove equivalences under the homomorphic equation (Equation E7). Secondly, our theorem states ballot secrecy for any number n of participants and ProVerif cannot handle parametrised processes (see Paiola & Blanchet [61, 60] for some initial progress in this direction). We proceed by constructing a relation that relates $A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$ and $A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$, and all their successors, such that it satisfies the three properties of Definition 2. In particular, the two final frames (containing the result of the election) should be statically equivalent.

Definition 6 (Valid ballot)**.** *A term T is said to be a* valid ballot *in an election with ℓ candidates if $[\![\phi^{\mathsf{sol}}_{\ell,0}\{^{T}/_{y_{\mathsf{ballot}}}\}]\!] = \mathsf{true}$.*

By definition, the bulletin board accepts only valid ballots. A key step to proving static equivalence is to show that any valid ballot submitted to the bulletin board by the environment is "equivalent" to a term of the form $\langle\!|\mathsf{penc}(z_{\mathsf{pk}}, N_1, M_1), \ldots, \mathsf{penc}(z_{\mathsf{pk}}, N_\ell, M_\ell), S_1, \ldots, S_{\ell+1}|\!\rangle$, where $\{^{M_1}/_{x_1^{\mathsf{vote}}}, \ldots, ^{M_\ell}/_{x_\ell^{\mathsf{vote}}}\}$ is a candidate substitution. This allows us to deduce that the election outcome produced by $A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$ is exactly the same as in $A^{\phi}_{\ell,n}[V_\ell\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V_\ell\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$. We can then conclude the proof of Theorem 1 by showing that the partial decryptions and the encrypted ballots of honest voters do not leak any extra information to the adversary. The full proof appears in Appendix B.

5.4 Limitations

The limitations of our model, which we introduced to simplify the presentation and proof, are detailed below; we believe a full security proof should follow using similar reasoning. We make use of a (standard) definition of ballot secrecy which is limited to elections with two honest voters [49, 30, 31]. In addition, the definition of ballot secrecy does not consider parallel composition of protocol executions and we therefore recommend using distinct keys for each election (although we believe it should be sufficient to include an election identifier – for example, the election fingerprint – in the challenge hashes included within signatures of knowledge, similar to the methodology in Section 4.2). The administrative process $A^{\phi}_{\ell,n}$ enforces an ordering on voters (namely, the voter using private channel a_1 must vote first, followed by the voter using private channel a_2, and then any remaining voters – controlled by the adversarial environment – can vote); this limitation could be overcome by parameterising $A^{\phi}_{\ell,n}$ with the channel names to restrict and by a minor unification of the bulletin process $BB^{\phi}_{\ell,n}$, however, this generalisation is of limited interest and would come at the cost of over-complicating the proof. In addition, the administrative process $A^{\phi}_{\ell,n}$ does not permit revoting. The signature and equational theory do not capture low-level technical details surrounding the correct construction of public keys; in particular, we do not use signatures of knowledge to verify correct key construction. We also omit signatures of knowledge that demonstrate correct construction of partial decryptions. Finally, we offer the usual caveat to formal analysis and acknowledge that our result does not imply the absence of real-world attacks (see, for example, [69, 2, 3, 81, 82]). It may, therefore, be possible to modify the ballot in a way that would not be captured by our analysis. (In particular, it is important to notice that the scheme used for signatures of knowledge is not provably non-malleable.) We partly overcome these limitations in our complimentary work [15] by presenting a variant of Helios that is provably secure in a cryptographic setting.

6 Attacks against other schemes

This section demonstrates that the absence of ballot independence can be exploited in other electronic voting protocols to violate privacy. In particular, we demonstrate replay attacks against schemes by Sako & Kilian [70] and Schoenmakers [72], both of which were presented at CRYPTO, and we show that the malleable cryptographic scheme adopted by Lee *et al.* [55] can be exploited to launch attacks. In addition, perhaps contrary to intuition, we will argue that no general relationships exist between independence and privacy properties.

6.1 Exploiting replays in the protocol by Sako & Kilian

The Sako & Kilian [70] electronic voting scheme capitalises upon advances in cryptography to improve the Banaloh & Yung protocol [14]. The scheme is interesting because it was one of the first electronic voting protocols to adopt the

Fiat-Shamir heuristic to derive non-interactive proofs (this evolution was key for the development of end-to-end verifiable electronic voting systems). However, we will show that the application of the Fiat-Shamir heuristic compromises ballot secrecy. In particular, the interactive nature of zero-knowledge proofs guarantees ballot independence; whereas, non-interactive proofs, derived using the Fiat-Shamir heuristic, do not assure independence. This can be exploited by a replay attack to violate ballot secrecy.

6.1.1 Protocol description

The scheme is based upon a pair of *partially compatible homomorphic encryption* functions, that is, a pair of functions f_1, f_2 over $\mathbb{Z}_q$, where q is prime, such that for all $i, j \in \{1, 2\}$ the following properties are satisfied:

- $f_i(x + y) = f_i(x) \cdot f_i(y)$, where $x, y \in \mathbb{Z}_q$
- Distributions $(f_i(x), f_j(y))$ and $(f_i(x), f_j(x))$ are computationally indistinguishable, where x and y are chosen uniformly in $\mathbb{Z}_q$.

The voting protocol is defined for $m \in \mathbb{N}$ voters as follows.

Setup. Talliers $\mathcal{T}$ and $\mathcal{T}'$ publish public keys k and k' for a public key encryption scheme E (which need not be homomorphic).

Voting. Given vote $v_i \in \{-1, 1\}$, the voter generates nonces $x_i, x_i' \in \mathbb{Z}_q$ such that $v_i = x_i + x_i'$ and constructs her ballot as follows:

$$\begin{array}{rcl} Y_i & = & f_1(x_i) \\ Y_i' & = & f_2(x_i') \\ Z_i & = & E(k, x_i) \\ Z_i' & = & E(k', x_i') \end{array}$$

In addition, the voter is required to prove $x_i + x_i' \in \{1, -1\}$ in zero-knowledge. However, to avoid an interactive proof, the Fiat-Shamir heuristic is applied to derive a signature of knowledge σ_i. (For brevity we omit the construction of σ_i, see [70, Figure 1] for details.)

Tallying. Given ballots $Y_1, Y_1', Z_1, Z_1', \sigma_1, \ldots, Y_n, Y_n', Z_n, Z_n', \sigma_n$, tallier $\mathcal{T}$ decrypts each Z_i to recover $\hat{x}_i$ and checks $Y_i = f_1(\hat{x}_i)$, similarly, tallier $\mathcal{T}'$ decrypts Z_i' to recover $\hat{x}_i'$ and checks $Y_i' = f_1(\hat{x}_i')$; the talliers also check the signature of knowledge σ_i. The talliers publish $V = \sum_{i=1}^{m} \hat{x}_i$ and $V' = \sum_{i=1}^{m} \hat{x}_i'$, and the result is $T = V + V'$, which can be verified by checking $f_1(V) = \prod_{i=1}^{m} Y_i$ and $f_2(V') = \prod_{i=1}^{m} Y_i'$.

6.1.2 Attacking ballot secrecy

We show that the voting protocol by Sako & Kilian does not satisfy ballot secrecy, by presenting a replay attack which allows an adversary to reveal a voter's vote. Intuitively, an adversary may observe the ballot posted by a particular voter and recast this ballot by corrupting dishonest voters. The multiple occurrences of the voter's ballot will leak information in the tally and the adversary can exploit this knowledge to violate the voter's privacy. An informal description of the attack will now be presented in the case of three eligible voters.

Let us consider an election with three eligible voters who have identities id_1, id_2 and id_3. Suppose that voters id_1, id_2 are honest and id_3 is a dishonest voter controlled by the adversary. Further assume that the adversary has observed the ballot

$$Y_k, Y'_k, Z_k, Z'_k, \sigma_k$$

being cast by the voter whose privacy will be compromised.

Exploiting the absence of ballot independence. As shown by Gennaro [43], an adversary can replay the ballot $Y_k, Y'_k, Z_k, Z'_k, \sigma_k$, thereby violating ballot independence. (The violation of ballot independence is due to the adversary's ability to cast the *same* vote as the honest voter.) Since the ballot was constructed by an honest voter, it is trivial to see that it will be considered valid by the talliers. We will now show how the lack of ballot independence can be exploited to violate privacy.

Violating privacy. The bulletin board will be constructed as follows

$$Y_1, Y'_1, Z_1, Z'_1, \sigma_1, Y_2, Y'_2, Z_2, Z'_2, \sigma_2, Y_k, Y'_k, Z_k, Z'_k, \sigma_k, V, V'$$

where $k \in \{1, 2\}$, $V = x_1 + x_2 + x_k$ and $V' = x'_1 + x'_2 + x'_k$. It follows from the protocol description that $v_i = x_i + x'_i$, where $i \in \{1, 2, k\}$, and the result $T = V + V' = v_1 + v_2 + v_k$. Since there will be at least two votes for the candidate voter id_k voted for, the voter's vote can be revealed: if $T \geq 2$, then $v_k = 1$; otherwise $v_k = -1$. It follows that the voter's privacy has been compromised; moreover, the vote of the remaining honest voter is $T - 2 \cdot v_k$.

6.1.3 Independence and the Fiat-Shamir heuristic

The interactive nature of zero-knowledge proofs guarantee independence; by comparison, non-interactive proofs, derived using the Fiat-Shamir heuristic, do not assure independence. As a consequence, application of the Fiat-Shamir heuristic may compromise the security of cryptographic protocols and we have shown how application of the heuristic erodes privacy in the electronic voting scheme by Sako & Kilian. This demonstrates that the use of the Fiat-Shamir heuristic requires some care and highlights the necessity for thorough security analysis.

6.1.4 Generalising replay attacks

The replay attack against the voting protocol by Sako & Kilian can be generalised to other schemes where an adversary can observe a ballot cast by a particular voter and replay this ballot verbatim. In particular, the voting protocol by Schoenmakers [72] fits this description.

Exploiting replays in the protocol by Schoenmakers. The electronic voting protocol by Schoenmakers [72] is based upon [28, 29]. The scheme explicitly aims to provide efficient small-scale elections (for example, boardroom elections) and, given that our attack is particularly well suited to small-scale elections, we find it interesting to study the possibility of violating ballot secrecy in this setting. Ballot independence is not provided [72, §5] and we exploit privacy using a replay attack. The attack description is straightforward and follows immediately from our discussion; accordingly, we omit the details and refer the interested reader to our technical report [77, §3].

6.1.5 Possible solutions: Weeding duplicate ballots

Our verbatim replay attacks against the voting protocols by Sako & Kilian and Schoenmakers exploit the possibility of replaying a voter's ballot without detection. We believe it should be sufficient for the election officer to reject any duplicate ballots to ensure ballot secrecy, alternatively, the *binding ballots to voters* solution (Section 4.2) may also be suitable, although proving the security of these solutions remains an open problem.

6.2 Exploiting malleability in the protocol by Lee *et al.*

The Lee *et al.* [55] electronic voting scheme adopts an offline tamper-resistant hardware device to ensure receipt freeness; more precisely, the hardware device takes an ElGamal encrypted vote as input and outputs a re-encrypted ciphertext, this prevents a voter proving how she voted by reconstruction as she does not know the nonce introduced for re-encryption. In addition, the hardware device provides a Designated Verifier Proof of re-encryption, thereby allowing the voter to verify that the device behaved correctly. The device is assumed to be offline and, hence, communication between the voter and the device is assumed to be untappable.

6.2.1 Background: Multiplicative homomorphic ElGamal

The scheme uses multiplicative homomorphic ElGamal, rather than the additive variant presented in Section 2.1. The operations for key generation, homomorphic combination and re-encryption are standard; albeit, the result of homomorphic combination is the multiplication of plaintexts, rather than the addition of plaintexts. We recall the operations for encryption and decryption below.

Encryption. Given a message m and a public key h, select a random nonce $r \in_R \mathbb{Z}_q^*$ and derive the ciphertext $(a, b) = (g^r \bmod p,\ m \cdot h^r \bmod p)$.

Distributed decryption. Given a ciphertext (a, b), each trustee $i \in n$ computes the partial decryption $k_i = a^{x_i}$. The plaintext $m = b/(k_1 \cdot \ldots \cdot k_n) \bmod p$.

The application of these primitives to derive the scheme by Lee *et al.* will be discussed in the next section.

6.2.2 Protocol description

An election is created by naming an election officer, selecting a set of mixers, and choosing a set of trustees. The trustees generate a distributed public key pair and the election officer publishes the public key on the bulletin board. (For robustness, threshold ElGamal may be used; we omit these details for brevity.) The election officer also publishes the candidate list, the public keys of eligible voters, and the public keys of the tamper-resistant hardware devices. Informally, the steps that the participants take during an election are as follows.

1. The voter constructs an ElGamal ciphertext (a, b) containing her vote v and sends the ciphertext to her tamper-resistant hardware device.

2. The hardware device re-encrypts the voter's ciphertext to produce (a', b') and computes a Designated Verifier Proof of re-encryption τ. The device also derives a signature σ on the re-encryption. The hardware device returns $(a', b'), \sigma, \tau$ to the voter.

3. If the signature and proof are valid, then the voter generates a signature σ' on the message σ using her private key. The voter submits her ballot $(a', b'), \sigma, \sigma'$ to the bulletin board.

4. Individual voters can check that their ballots appear on the bulletin board and can be assured that the ciphertext (a', b') contains their vote v by verifying the Designated Verifier Proof τ.

5. Voters and observers can check that ballots were cast by registered voters by verifying signatures σ', and are assured that each voter cast at most one ballot by checking that no voter signed two values. In addition, voters and observers should verify signatures σ for receipt freeness.

6. After some predefined deadline, valid ballots (that is, ballots associated with valid signatures σ and σ') are submitted to the mixers. Anyone can check that mixing is performed correctly.

7. Each of the trustees publishes a partial decryption for every ciphertext output by the mix. Anyone can verify these proofs.

8. The election officer decrypts each ciphertext and publishes the election result. Anyone can check these decryptions.

See Lee *et al.* [55] for further details.

6.2.3 Attacking ballot secrecy

We show that the voting protocol by Lee *et al.* [55] does not satisfy ballot secrecy by recalling the replay attack by Dreier, Lafourcade & Lakhnech [36] that exploits malleability to reveal a voter's vote. Intuitively, an adversary may identify a voter's encrypted vote on the bulletin board, since it is signed by the voter. This ciphertext can be submitted to a tamper-resistant hardware device (possibly after re-encryption) and the device will return $(\hat{a}, \hat{b}), \hat{\sigma}, \hat{\tau}$; the ballot $(\hat{a}, \hat{b}), \hat{\sigma}, \hat{\sigma}'$ can then be submitted by the adversary to the bulletin board, where $\hat{\sigma}'$ is a signature on $\hat{\sigma}$ constructed by a registered voter under the adversary's control. As explained in Section 6.1.2, the multiple occurrences of the voter's ballot will leak information in the tally and the adversary can exploit this knowledge to violate the voter's privacy.

Variant exploiting homomorphic encryption. The adversary can exploit the homomorphic properties of ElGamal to avoid casting the *same* vote as an honest voter. In this variant, suppose the adversary wants to recover the vote from ballot $(a'_k, b'_k), \sigma_k, \sigma'_k$, the adversary derives the ciphertext $(c, d) = (a'_k, b'_k) \cdot (c', d')$, where (c', d') is an ElGamal ciphertext containing some message m. The adversary submits the ciphertext (c, d) to a tamper-resistant hardware device and the device will return $(\hat{c}, \hat{d}), \hat{\sigma}, \hat{\tau}$; the ballot $(\hat{c}, \hat{d}), \hat{\sigma}, \hat{\sigma}'$ can be submitted by the adversary to the bulletin board, where $\hat{\sigma}'$ is a signature on $\hat{\sigma}$ constructed by a registered voter. The output of the mix will include the adversaries re-encrypted ciphertext and the election officer will publish $m \cdot v$ on the bulletin board, where ciphertext (a'_k, b'_k) includes the vote v. This variant of the attack is particularly interesting because the replayed ballots are undetectable; in particular, weeding duplicate ballots would clearly not be sufficient to ensure privacy.

6.2.4 Privacy in the variant by Küsters & Truderung

Independently, the scheme by Lee *et al.* is vulnerable to a forced abstention attack [51, §6.2], whereby the voter votes for a candidate not included on the candidate list. This can be trivially witnessed since casting a vote for some random nonce will result in a unique occurrence of that nonce being published on the bulletin board by the election officer. Accordingly, Küsters & Truderung [51, §6.3] propose a variant of the protocol by Lee *et al.* which they claim satisfies coercion resistance. In the revised scheme the voter must prove to her tamper-resistant hardware device that her ciphertext contains a valid vote. However, our attack is still valid under one of the following assumptions: 1) there exists a tamper-resistant hardware device which does not check the proof that the voter's ciphertext is correctly formed; 2) there exists a signing key which has been extracted from a hardware device; or 3) the election officer publishes the public part of a signing key where the private part is known to the adversary. The attack follows immediately from our original description under assumption

Figure 5 Helios administrator that preserves independence but not privacy

Given the number of voters $n \geq 2$ the administration process $\overline{A}_n^{\phi^{\mathsf{sol}}}$ is defined below, where process T is presented in Figure 4.

$$\begin{aligned}
\overline{A}_n^{\phi^{\mathsf{sol}}} &= \nu\, sk_T, a_1, a_2, d\,.\,(_ \mid \overline{BB}_n^{\phi^{\mathsf{sol}}} \mid !T \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}) \\
\\
\overline{BB}_n^{\phi^{\mathsf{sol}}} &= a_1(y_1)\,.\,\overline{c}\langle y_1\rangle\,.\,a_2(y_2)\,.\,\overline{c}\langle y_2\rangle\,. \\
&\quad a_3(y_3)\,.\,\text{if } \phi^{\mathsf{sol}}\{^{y_3}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
&\quad \cdots\ a_n(y_n)\,.\,\text{if } \phi^{\mathsf{sol}}\{^{y_n}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
&\quad \overline{d}\langle(\!|\pi_1(y_1), \pi_2(y_1)|\!)\rangle\,.\,d(z_1)\,.\,\overline{c}\langle z_1\rangle\,. \\
&\quad \cdots\,.\,\overline{d}\langle(\!|\pi_1(y_n), \pi_2(y_n)|\!)\rangle\,.\,d(z_n)\,.\,\overline{c}\langle z_n\rangle
\end{aligned}$$

1 and, under assumptions 2 or 3, the adversary signs the encrypted ciphertext without using the tamper-resistant hardware device.

6.3 Independence and privacy are unrelated properties

Intuitively, these attacks may be suggestive of a general relationship between independence and privacy properties, however, we shall now present examples that suggest independence does not imply privacy and vice-versa.

A protocol with independence but no privacy. Consider a variant of the fixed Helios voting scheme in which each of the trustees publish a partial decryption of individual ciphertexts (rather than a partial decryption of the homomorphically combined ciphertexts, that is, the encrypted tally). Intuitively, this variant preserves ballot independence but does not satisfy ballot secrecy, since the partial decryptions allow votes to be recovered from ballots and the link between a voter and her ballot is known. Formally, this variant is captured by modelling the Helios administrator process as $\overline{A}_n^{\phi^{\mathsf{sol}}}$, defined in Figure 5. The violation of ballot secrecy can be witnessed since

$$\overline{A}_2^{\phi^{\mathsf{sol}}}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau] \not\approx_l \overline{A}_2^{\phi^{\mathsf{sol}}}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$$

where $\sigma = \{^{\mathsf{zero}}/_{x_1^{\mathsf{vote}}}, ^{\mathsf{one}}/_{x_2^{\mathsf{vote}}}\}$ and $\tau = \{^{\mathsf{one}}/_{x_1^{\mathsf{vote}}}, ^{\mathsf{zero}}/_{x_2^{\mathsf{vote}}}\}$. Similarly, a further variant of the fixed Helios scheme in which each of the trustees publishes their private key at the end of the voting phase, rather than a partial decryption of the encrypted tally, also satisfies independence but not ballot secrecy.

A protocol with privacy but no independence. Consider a voting scheme in which each voter broadcasts their vote on an anonymous communication channel. Formally, the voter is modelled by the process $P = \overline{c}\langle x_{\mathsf{vote}}\rangle$, where variable x_{vote} is parametrised by the voter's vote. For ballot secrecy it is sufficient to show $P\{^M/_{x_{\mathsf{vote}}}\} \mid P\{^N/_{x_{\mathsf{vote}}}\} \approx_l P\{^N/_{x_{\mathsf{vote}}}\} \mid P\{^M/_{x_{\mathsf{vote}}}\}$ for all ground

terms M and N; this result trivially holds by structural equivalence and hence the scheme satisfies ballot secrecy. However, independence is intuitively violated in this setting, because an adversary may observe the voting system and replay a previously cast vote, that is, an adversary can cast the same vote as another voter (without knowing which voter). In addition, it follows that early results are available in this scheme.

We also expect some published electronic voting schemes based upon blind signatures to satisfy ballot secrecy but not independence; in particular, a more realistic example of a protocol that satisfies this property is the protocol by Fujioka, Okamoto & Ohta [42] under the assumption that duplicates are not rejected. Indeed, independence can be violated by a verbatim replay of the signed committed vote.

We believe the existence of a weaker property: *privacy and authenticated ballots implies independence*, where the term *authenticated ballot* means the link between an arbitrary ballot and associated voter is known. Informally, this can be witnessed as follows: suppose a system satisfies privacy and authenticated ballots but not independence, it follows that an adversary can identify a voter's ballot and, since there is no independence, replay that ballot; privacy is then violated, as we have shown in this article, hence deriving a contradiction.

In this article, we cannot make any definitive mathematical statements about the relationship between independence and privacy properties, because independence has not been formally defined; however, we hope these examples provide some insight into the relationships we expect.

7 Conclusion and further discussion

This article identifies a vulnerability in the Helios 2.0 electronic voting protocol which can be used to violate ballot secrecy. Critics may argue that an attack is unrealistic due its high cost; indeed, in some cases, the attack may change the outcome of an election (that is, the votes introduced for the purposes of violating privacy may swing the result), and large scale privacy invasions would be expensive in terms of the required number of dishonest voters. However, if the views of these critics are to be entertained, then we must revise the standard definitions of ballot secrecy in the literature (for example, [49, 30, 9]) because Helios cannot satisfy them. Furthermore, we believe all voters should be considered equally and, hence, the preservation of ballot secrecy should be universal. But, for elections using Helios, our case study demonstrates the contrary: in French legislative elections a coalition of voters can gain some information about a voter's vote in an arbitrary polling station and, moreover, if the number of voters registered at a particular polling station is small (for example, in a rural setting), then a voter's privacy can be violated by a few dishonest voters. It follows that privacy of individual voters can be compromised by a few dishonest voters and, accordingly, we believe our attack is significant. To address the problem, we have introduced a variant of the Helios protocol which has been shown

to satisfy definitions of ballot secrecy in the applied pi calculus and in our complimentary work [15] we present a security proof in the cryptographic setting. The vulnerability in Helios has been acknowledged by Adida & Pereira [6, 8] and they have scheduled a fix for future Helios releases. We have also shown that the absence of ballot independence can be similarly exploited in other electronic voting protocols to violate privacy; in particular, we demonstrate verbatim replay attacks against the schemes by Sako & Kilian [70] and Schoenmakers [72], and we show that the malleable cryptographic scheme adopted by Lee *et al.* [55] can be exploited to replay a voter's ballot or a variant of it, thereby violating ballot secrecy. In addition, we argue that independence does not imply privacy and vice-versa. Finally, all of the vulnerabilities in this article have been acknowledged by the protocol authors, with the exception of Schoenmakers; in particular, Adida & Pereira have acknowledged the vulnerability in Helios [6, 8] and they have scheduled a fix for future Helios releases.

Acknowledgements

We are grateful to Ben Adida and Olivier Pereira for their constructive comments, and hope this research will enhance future Helios releases. Discussion with Mark D. Ryan helped clarify the presentation of this article and Ben Adida informed us that Douglas Wikström is the contemporaneous discoverer of the Helios attack. David Bernhard gave useful feedback on the variants of our attack against Helios and Christian Cachin highlighted Josh Benaloh's related work. The research leading to these results was performed as part of the ProSecure project which is funded by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865, and the ANR-07-SeSur-002 AVOTÉ project.

A Signatures of knowledge

Helios is reliant on signatures of knowledge to ensure secrecy and integrity of the ElGamal scheme, and to ensure voters encrypt valid votes. This appendix presents suitable cryptographic primitives. Let $\mathcal{H}$ denote a hash function. In Helios, $\mathcal{H}$ is defined to be SHA-256.

A.1 Knowledge of discrete logs

Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating knowledge of a discrete logarithm $h = \log_g g^x$ can be derived, and verified, as defined by [18, 17, 71].

Sign. Given x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witness $g' = g^w \bmod p$, challenge $c = \mathcal{H}(g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify. Given h and signature g', s, check $g^s \equiv g' \cdot h^c \pmod{p}$, where $c = \mathcal{H}(g') \bmod q$.
A valid proof asserts knowledge of x such that $x = \log_g h$; that is, $h \equiv g^x \bmod p$.

A.2 Equality between discrete logs

Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating equality between discrete logarithms $\log_f f^x$ and $\log_g g^x$ can be derived, and verified, as defined by [63, 19].

Sign. Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $f' = f^w \bmod p$ and $g' = g^w \bmod p$, challenge $c = \mathcal{H}(f', g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify. Given f, g, h, k and signature f', g', s, check $f^s \equiv f' \cdot h^c \pmod{p}$ and $g^s \equiv g' \cdot k^c \pmod{p}$, where $c = \mathcal{H}(f', g') \bmod q$.
A valid proof asserts $\log_f h = \log_g k$; that is, there exists x, such that $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$. This signature of knowledge scheme can be extended to a disjunctive proof of equality between discrete logs (see below).

For our purposes, given a ciphertext (a, b), each trustee would derive a signature on g, a, x_i, where x_i is the trustee's private key share. The ith trustee's signature g_i', a_i', c_i, s_i would be verified with respect to g, a, h_i, k_i, where h_i is the trustee's share of the public key and k_i is the trustee's partial decryption; that is, the proof asserts $\log_g h_i = \log_a k_i$, as required for integrity of decryption.

A.3 Disjunctive proof of equality between discrete logs

Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating that a ciphertext (a, b) contains either 0 or 1 (without revealing which), can be constructed by proving that either $\log_g a = \log_h b$ or $\log_g a = \log_h b/g^m$; that is, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms [27, 73]. Observe for a valid ciphertext (a, b) that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$; hence the former disjunct $\log_g g^r = \log_h h^r \cdot g^m$ is satisfied when $m = 0$, and the latter $\log_g g^r = \log_h (h^r \cdot g^m)/g^m$ when $m = 1$.

This technique is generalised by [7] to allow a signature of knowledge demonstrating that a ciphertext (a, b) contains message m, where $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$ for some system parameters $\mathsf{min}, \mathsf{max} \in \mathbb{N}$. Formally, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms can be derived, and verified, as follows [7, 27, 73].

Sign. Given ciphertext (a, b) such that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$, where plaintext $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$. For all $i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}$, compute challenge $c_i \in_R \mathbb{Z}_q^*$, response $s_i \in_R \mathbb{Z}_q^*$ and witnesses $a_i = g^{s_i}/a^{c_i} \bmod p$ and $b_i = h^{s_i}/(b/g^i)^{c_i} \bmod p$. Select a random

nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $a_m = g^w \bmod p$ and $b_m = h^w \bmod p$, challenge $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) - \sum_{i \in \{\mathsf{min},\ldots,m-1,m+1,\ldots,\mathsf{max}\}} c_i \pmod q$ and response $s_m = w + r \cdot c_m \bmod q$.

Verify. Given (a, b) and $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$, for each $\mathsf{min} \leq i \leq \mathsf{max}$ check $g^{s_i} \equiv a_i \cdot a^{c_i} \pmod p$ and $h^{s_i} \equiv b_i \cdot (b/g^i)^{c_i} \pmod p$. Finally, check $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) \equiv \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \pmod q$.
A valid proof asserts that (a, b) is a ciphertext containing the message m such that $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$.

B Proof of Theorem 1

B.1 Preliminaries

Before commencing our proof, let us first introduce some useful lemmas for the applied pi calculus.

Lemma 3. *Given frames φ, ψ, ground term M and variable $x \notin \mathrm{dom}(\varphi) \cup \mathrm{dom}(\psi)$, we have $\varphi \approx_s \psi$ iff $\varphi \mid \{^M/_x\} \approx_s \psi \mid \{^M/_x\}$.*

Lemma 4. *Given frames φ, ψ, terms M, N, and a variable $x \notin \mathrm{dom}(\varphi) \cup \mathrm{dom}(\psi)$, such that $\varphi = \nu\, \tilde{m}.\sigma$ and $\psi = \nu\, \tilde{n}.\tau$ for some names $\tilde{m}$, $\tilde{n}$ and substitutions σ, τ, we have $\nu\, \tilde{m}.(\sigma \mid \{^M/_x\}) \approx_s \nu\, \tilde{n}.(\tau \mid \{^N/_x\})$ implies $\varphi \approx_s \psi$.*

The proofs of these lemmas are straightforward.

The following lemma shows when static equivalence implies the same branching behaviour for conditionals.

Lemma 5. *Given extended processes $A \equiv C[\text{if } M = N \text{ then } P \text{ else } Q]$ and $B \equiv C'[\text{if } M = N \text{ then } P' \text{ else } Q']$ such that $A \approx_s B$, $(\mathrm{bn}(C) \cup \mathrm{bn}(C')) \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$, $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C)$ and $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C')$, for some closing evaluation context C, C', terms M, N and processes P, P', Q, Q', then $A \to C[P]$ iff $B \to C'[P']$ and $A \to C[Q]$ iff $B \to C'[Q']$.*

Proof. Suppose $A \equiv C[\text{if } M = N \text{ then } P \text{ else } Q]$ and $B \equiv C'[\text{if } M = N \text{ then } P' \text{ else } Q']$ such that $A \approx_s B$, $(\mathrm{bn}(C) \cup \mathrm{bn}(C')) \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$, $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C)$ and $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C')$, for some closing evaluation context C, C', terms M, N and processes P, P', Q, Q'. Further suppose $\varphi(C[\text{if } M = N \text{ then } P \text{ else } Q]) = \nu\, \tilde{m}.\sigma$ and $\varphi(C'[\text{if } M = N \text{ then } P' \text{ else } Q']) = \nu\, \tilde{n}.\tau$, for some names $\tilde{m}$ and $\tilde{n}$. By Lemma 6 we have $\nu\, \tilde{m}.\sigma \approx_s \nu\, \tilde{n}.\tau$, because static equivalence is closed under structural equivalence. Moreover, by the definition of static equivalence, for all terms U, V such that $(\tilde{m} \cup \tilde{n}) \cap (\mathrm{fn}(U) \cup \mathrm{fn}(V)) = \emptyset$, we have $U\sigma =_E V\sigma$ iff $U\tau =_E V\tau$.

Let us first show $A \to C[P]$ iff $B \to C'[P']$. For the $\Rightarrow$ implication, suppose $A \to C[P]$. Since $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C)$, it must be the case that $M\sigma =_E N\sigma$. We have $\tilde{m} \cup \tilde{n} \subseteq \mathrm{bn}(C) \cup \mathrm{bn}(C')$ by definition of the function φ, and we derive $(\tilde{m} \cup \tilde{n}) \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$ because $(\mathrm{bn}(C) \cup \mathrm{bn}(C')) \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$; it

follows that $M\sigma =_E N\sigma$ is a special case of $U\sigma =_E V\sigma$. We derive $M\tau =_E N\tau$ from the implication $(U\sigma =_E V\sigma) \Rightarrow (U\tau =_E V\tau)$. It trivially follows that $B \equiv C'[\text{if } M\tau = M\tau \text{ then } P' \text{ else } Q']$, and by closure of internal reduction under structural equivalence we derive $B \to C'[P']$. The $\Leftarrow$ implications follows by symmetry.

We will now show $A \to C[Q]$ iff $B \to C'[Q']$. For the $\Rightarrow$ implication, suppose $A \to C[Q]$. It must be the case that $M\sigma \neq_E N\sigma$ and, as before, we derive $M\tau \neq_E N\tau$. It trivially follows that $B \equiv C'[\text{if } M\tau = N\tau \text{ then } P' \text{ else } Q']$, and since $\mathrm{fv}(M) \cup \mathrm{fv}(N) \subseteq \mathrm{dom}(C')$ we are assured that terms $M\tau, N\tau$ are ground; by closure of internal reduction under structural equivalence we derive $B \to C'[Q']$. The $\Leftarrow$ implications follows by symmetry. □

This result can naturally be extended to formula. Given ϕ, let us denote the set of free names, respectively variables, in ϕ as $\mathrm{fn}(\phi)$, respectively $\mathrm{fv}(\phi)$.

Corollary 1. *Given extended processes $A \equiv C[\text{if } \phi \text{ then } P \text{ else } Q]$ and $B \equiv C'[\text{if } \phi \text{ then } P' \text{ else } Q']$ such that $A \approx_s B$, $(\mathrm{bn}(C) \cup \mathrm{bn}(C')) \cap \mathrm{fn}(\phi) = \emptyset$, $\mathrm{fv}(\phi) \subseteq \mathrm{dom}(C)$ and $\mathrm{fv}(\phi) \subseteq \mathrm{dom}(C')$, for some closing evaluation context C, C', formulae ϕ and processes P, P', Q, Q', then $A \to C[P]$ iff $B \to C'[P']$ and $A \to C[Q]$ iff $B \to C'[Q']$.*

We conclude this subsection with a useful result stated by Abadi & Fournet [1].

Lemma 6. *Static equivalence is closed by structural equivalence.*

B.2 Notations and Definitions

For the remainder of this article, let ℓ be some number of candidates, $n \geq 2$ be some number of voters, and σ and σ' be candidate substitutions.

B.2.1 Notations

We introduce the following notations for all $1 \leq i \leq n$ and $1 \leq j \leq \ell$:

$$
\begin{aligned}
tally_j &= \pi_j(y_1) * \cdots * \pi_j(y_n) \\
partial_j &= \mathsf{partial}(sk_T, tally_j) \\
result_j &= \mathsf{dec}(partial_j, tally_j) \\
ciph_{i,j} &= \mathsf{penc}(z_{\mathsf{pk}}, r_{i,j}, x^{\mathsf{vote}}_{i,j}) \\
spk_{i,j} &= \mathsf{spk}(z_{\mathsf{pk}}, r_{i,j}, x^{\mathsf{vote}}_{i,j}, ciph_{i,j}) \\
\widehat{spk}_i &= \mathsf{spk}(z_{\mathsf{pk}},\ r_{i,1} \circ \cdots \circ r_{i,\ell},\ x^{\mathsf{vote}}_{i,1} + \cdots + x^{\mathsf{vote}}_{i,\ell},\ ciph_{i,1} * \cdots * ciph_{i,\ell}) \\
ballot_i &= (\!| ciph_{i,1}, \ldots, ciph'_{i,\ell}, spk_{i,1}, \ldots, spk'_{1,\ell}, \widehat{spk}_i |\!) \\
\tau_L &= \{{}^{M}/_{x^{\mathsf{vote}}_{1,i}} \mid \text{for all } 1 \leq i \leq \ell \text{ such that} \{{}^{M}/_{x^{\mathsf{vote}}_{i}}\} \in \sigma\} \\
&\quad \cup \{{}^{N}/_{x^{\mathsf{vote}}_{2,i}} \mid \text{for all } 1 \leq i \leq \ell \text{ such that } \{{}^{N}/_{x^{\mathsf{vote}}_{i}}\} \in \sigma'\} \\
\tau_R &= \{{}^{N}/_{x^{\mathsf{vote}}_{1,i}} \mid \text{for all } 1 \leq i \leq \ell \text{ such that} \{{}^{N}/_{x^{\mathsf{vote}}_{i}}\} \in \sigma'\} \\
&\quad \cup \{{}^{M}/_{x^{\mathsf{vote}}_{2,i}} \mid \text{for all } 1 \leq i \leq \ell \text{ such that } \{{}^{M}/_{x^{\mathsf{vote}}_{i}}\} \in \sigma\}
\end{aligned}
$$

B.2.2 Definitions

Given $N_3, \ldots, N_k$ terms such that $\mathrm{fv}(N_j) \subseteq \{z_{\mathsf{pk}}, y_1, \ldots, y_{j-1}\}$, we define

$$\sigma_{\tilde{N_k}} = \{{}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}, {}^{N_j}/_{y_j} \mid j \in \{3, \ldots, k\}\}$$

Given an integer $k \in \mathbb{N}^+$ and a term N, we define N^k (resp. $k.N$) to be $N \circ \cdots \circ N$ (resp. $N + \cdots + N$) where N is replicated k times.

We associate to the equational theory E a rewriting system $\mathcal{R}_E$ by orienting the Equations E1,E2 and E5 to E9 from left to right. We denote by E' the equational theory that asserts functions $+, *, \circ$ are commutative and associative in addition to Equations E3 and E4. $\mathcal{R}_E$ modulo E' forms a convergent rewriting system (modulo E'). We denote by $u \to_E v$ (or often simply $u \to v$) if u modulo E' can be rewritten to v modulo E', using $\mathcal{R}_E$. We denote by $u \downarrow$ a normal form of u modulo E'.

We will say that a term M is *free* w.r.t. a set of names $\tilde{n}$ if it does not contain any name of $\tilde{n}$. We simply say that a term is free when the set of names is clear from the context (typically free w.r.t. to the restricted names of a frame).

B.3 Some useful lemmas

We prove some useful results about our definitions and notations. We first show that ballots accepted by the bulletin board must have a particular form due to the checks performed by $\phi^{\mathsf{sol}}_{\ell,\bar{n}}$.

Lemma 7. *Let ℓ be a number of candidates, $\bar{n} \geq 2$ be an integer, and M be term free w.r.t. $r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}$ and such that $\mathrm{fv}(M) \subseteq \{z_{\mathsf{pk}}, y_1, y_2\}$. Let substitution $\tau \in \{\tau_L, \tau_R\}$ and substitution $\sigma = \{{}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}, {}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}\}$. If $[\![\phi^{\mathsf{sol}}_{\ell,\bar{n}}\{{}^{M}/_{y_{\mathsf{ballot}}}\}\sigma\tau]\!] = \mathsf{true}$, then there exists a term*

$$M' = (\!|\mathsf{penc}(z_{\mathsf{pk}}, N_1, M_1), \ldots, \mathsf{penc}(z_{\mathsf{pk}}, N_\ell, M_\ell), S_1, \ldots, S_{\ell+1}|\!)$$

for some terms $M_1, \ldots, M_\ell, N_1, \ldots, N_\ell, S_1, \ldots, S_{\ell+1}$ such that $M\sigma\tau =_E M'\sigma\tau$, M' is free w.r.t. $r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}$, $\mathrm{fv}(M') \subseteq \{z_{\mathsf{pk}}, y_1, y_2\}$, and $\{{}^{M_1}/_{x_1^{\mathsf{vote}}}, \ldots, {}^{M_\ell}/_{x_\ell^{\mathsf{vote}}}\}$ is a candidate substitution.

Proof. Let M, τ and σ be defined as in the Lemma, and suppose $[\![\phi^{\mathsf{sol}}_{\ell,\bar{n}}\{{}^{M}/_{y_{\mathsf{ballot}}}\}\sigma\tau]\!] = \mathsf{true}$. We say that a term N is a *minimal recipe* if it is minimal (in size) among the terms N' such that $N\sigma\tau =_E N'\sigma\tau$. It is easy to check by induction on the size of N that, whenever $N = f(N_1, \ldots, N_k)$ with $f \in \{\mathsf{dec}, \pi_j \mid 1 \leq j \leq \ell\}$ then either $N = \pi_j(x)$ for some j and variable x or $(N\sigma\tau)\downarrow = f((N_1\sigma\tau)\downarrow, \ldots, (N_k\sigma\tau)\downarrow)$ (*).

W.l.o.g. suppose M' is a minimal recipe such that $M\sigma\tau =_E M'\sigma\tau$ and M' is free w.r.t. $r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}$. Further suppose w.l.o.g. that M' is in normal form. We know $[\![\phi^{\mathsf{sol}}_{\ell,\bar{n}}\{{}^{M'}/_{y_{\mathsf{ballot}}}\}\sigma\tau]\!] = \mathsf{true}$. Thus it must be the case that $M'\sigma\tau$ is of the form $(\!|U_1, \ldots, U_\ell, V_1, \ldots, V_\ell, W|\!)$, where for $1 \leq j \leq \ell$ we have $U_j = \mathsf{penc}(\mathsf{pk}(sk_T), R_j, C_j)$, $C_j \in \{\mathsf{zero}, \mathsf{one}\}$ and $\{{}^{C_1}/_{x_1^{\mathsf{vote}}}, \ldots, {}^{C_\ell}/_{x_\ell^{\mathsf{vote}}}\}$ is

a candidate substitution. Due to the disequality tests in $\phi^{\mathsf{sol}}_{\ell,\tilde{n}}$, it must be the case that M' is of the form $(\!|T_1, \ldots, T_\ell, S_1, \ldots, S_\ell, Z|\!)$ and $T_j \notin \{\pi_k(y_i) \mid 1 \leq k \leq \ell\}$. We have $T_j\sigma\tau = \mathsf{penc}(A_j, B_j, C_j)$. Assume first that $T_j = \pi_k(T'_j)$. Due to (*), we must have T'_j variable, which is excluded by the fact that $T_j \notin \{\pi_k(y_i) \mid 1 \leq k \leq \ell\}$. Thus, due to the equational theory and (*), it must be the case that $T_j = \mathsf{penc}(K_j, N_j, M_j) * \prod_{1\leq k\leq \ell} \pi_k(y_1)^{\alpha_k} * \pi_k(y_2)^{\beta_k}$ where each component is optional and $\alpha_i \in \mathbb{N}$. By convention $\alpha_i = 0$ or $\beta_i = 0$ means that the component is skipped. Assume that one of the α_i or β_i is not null. Then $R_j = r \circ R'_j$ with $r \in \{r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\}$. Due to the tests in $\phi^{\mathsf{sol}}_{\ell,\tilde{n}}$, we know $V_j = \mathsf{spk}(\mathsf{pk}(sk_T), R_j, C_j, V'_j)$.

Let us show that V_j cannot be a signature of knowledge that appears in either $ballot_1\tau_L$ or $ballot_2\tau_L$. Assume (by contradiction) that V_j is a signature of knowledge that appears in either $ballot_1\tau_L$ or $ballot_2\tau_L$. Due to weeding, we cannot have $V_j = \mathsf{spk}_{i,k}$. Indeed, due to the equational theory, this would imply that U_j is equal to a previsouly received cyphertext, which is excluded by weeding. Thus we must have $V_j = \widehat{spk_i}$ for some $i \in \{1, 2\}$. Then $R_j = r_{i,1} \circ \cdots \circ r_{i,\ell}$. In that case, let us have a look at W. We know $W = \mathsf{spk}(\mathsf{pk}(sk_T), R_1 \circ \cdots \circ R_\ell, C_1 + \cdots + C_\ell, U_1 * \cdots * U_\ell)$. Thus W cannot be one of the signatures of knowledge that appear in $ballot_1\tau_L$ or $ballot_2\tau_L$ (the depth of $R_1 \circ \cdots \circ R_\ell$ is too big). Therefore (and due to the equational theory and minimality of Z), we must have $Z = \mathsf{spk}(Z^1, Z^2, Z^3, Z^4)$. Since $r_{i,1} \circ \cdots \circ r_{i,\ell}$ is not deducible, we cannot have $Z^2\sigma\tau =_E r_{i,1} \circ \cdots \circ r_{i,\ell} \circ R_1 \circ R_{j-1} \circ R_{j+1} \circ R_\ell$, contradiction.

We must have $S_j = \mathsf{spk}(S^1_j, S^2_j, S^3_j, S^4_j)$, since V_j cannot be one of the signatures of proof of knowledge that appear in $ballot_1\tau_L$ or $ballot_2\tau_L$, and due to the equational theory. Since r is not deducible, we cannot have $S^2_j\sigma\tau =_E r \circ R'_j$, contradiction. We therefore deduce that $T_j = \mathsf{penc}(K_j, N_j, M_j)$. Moreover, $K_j\sigma\tau =_E \mathsf{pk}(sk_T)$ implies $K_j = z_{\mathsf{pk}}$ and $M_j\sigma\tau =_E \mathsf{zero}$ or one implies $M_j \in \{\mathsf{zero}, \mathsf{one}\}$ due to the equational theory. Due to the validity check, we also deduce that $\{M_1/x^{\mathsf{vote}}_1, \ldots, M_\ell/x^{\mathsf{vote}}_\ell\}$ is a candidate subsitution. □

Lemma 8. *Let* $\phi_1 = \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\ .\ (\{ballot_1/x_1\} \mid \{ballot_2/x_2\} \mid \{\mathsf{pk}(sk_T)/z_{\mathsf{pk}}\})$. *We have* $\phi_1\tau_L \approx_s \phi_1\tau_R$.

Proof. First, we decompose ϕ_1 and consider $\phi = \nu\, \tilde{n}.\theta$, where $\tilde{n} = \{sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\}$ and $\theta = \{\mathsf{pk}(sk_T)/z_{\mathsf{pk}}\} \mid \left\{\{ciph_{i,j}/x_{ciph_{i,j}}\} \mid \{spk_{i,j}/x_{spk_{i,j}}\} \mid \{\widehat{spk_i}/x_{\widehat{spk_i}}\} \;\middle|\; i \in \{1, 2\} \wedge 1 \leq j \leq \ell\right\}$. It follows immediately that $\phi_1\tau_L \approx_s \phi_1\tau_R$ if and only if $\phi\tau_L \approx_s \phi\tau_R$.

Secondly, witness that the adversary can arbitrarily combine ciphertexts from the frame – namely, ciphertexts $ciph_{1,1}, ciph_{2,1}, \ldots, ciph_{1,\ell}, ciph_{2,\ell}$ – with ciphertexts in the frame or freshly constructed ciphertexts, we enrich the frame ϕ with any such combination of ciphertexts. Formally, for any $\alpha_j, \beta_j \in \mathbb{N}$ and terms P, R we define $C_{\alpha_1,\ldots,\alpha_\ell,\beta_1,\ldots,\beta_\ell,\alpha_4,P,R}$ as follows:

$$\mathsf{penc}(\mathsf{pk}(sk_T), R \circ \textstyle\bigcirc_{1\leq j\leq \ell} r^{\alpha_j}_{1,j} \circ r^{\beta_j}_{2,j}, P + \sum_{1\leq j\leq \ell} \alpha_j.x^{\mathsf{vote}}_{1,j} + \beta_j.x^{\mathsf{vote}}_{2,j})$$

We define the extended frame ϕ_e below.

$$\phi_e = \nu\, \tilde{n}.(\theta \mid \{{}^{C_{\alpha_1,\alpha_2,\alpha_3,\alpha_4,P,R}}/_{x_{\alpha_1,\alpha_2,\alpha_3,\alpha_4,P,R}} \mid \alpha_1,\alpha_2,\alpha_3,\alpha_4 \in \mathbb{N} \text{ and terms } P,R \text{ s.t. } (\mathrm{fn}(P)\cup\mathrm{fn}(R))\cap\tilde{n}=\emptyset, \mathrm{fv}(P,R)\subseteq \mathrm{dom}(\phi_e) \text{ with no cycle}\})$$

Note that ϕ_e is infinite. By Lemma 4, it is sufficient to show $\phi_e\tau_L \approx_s \phi_e\tau_R$. We introduce the following two claims.

Claim 1. *Let M be a term such that $\mathrm{fv}(M) \cap (\mathrm{fv}(\phi_e) \setminus \mathrm{dom}(\phi_e)) = \emptyset$ and $\mathrm{fn}(M) \cap \tilde{n} = \emptyset$. If $M\phi_e\tau \to U$ for some $\tau \in \{\tau_R, \tau_L\}$, then there exists N such that $U =_{E'} N\phi_e\tau$ and $M\phi_e\tau' \to N\phi_e\tau'$ for any $\tau' \in \{\tau_R, \tau_L\}$.*

Claim 2. *Let M, N be two terms such that $(\mathrm{fv}(M) \cup \mathrm{fv}(N)) \cap (\mathrm{fv}(\phi_e) \setminus \mathrm{dom}(\phi_e)) = \emptyset$ and $\mathrm{fn}(M,N) \cap \tilde{n} = \emptyset$. If $M\phi_e\tau =_{E'} N\phi_e\tau$ for some $\tau \in \{\tau_R, \tau_L\}$, then $M\phi_e =_{E'} N\phi_e$.*

The above claims allow the construction of our proof. Let M, N be two terms such that $\mathrm{fn}(M,N) \cap \tilde{n} = \emptyset$ and $M\phi_e\sigma_{\tilde{N_k}}\tau_L =_E N\phi_e\sigma_{\tilde{N_k}}\tau_L$. We assume (possibly by renaming) that $(\mathrm{fv}(M) \cup \mathrm{fv}(N)) \cap (\mathrm{fv}(\phi_e)\backslash\mathrm{dom}(\phi_e)) = \emptyset$. We have $M\phi_e\tau_L =_E N\phi_e\tau_L$. Thus $(M\phi_e\tau_L)\downarrow =_{E'} (N\phi_e\tau_L)\downarrow$. Applying repeatedly Claim 1, we deduce that there exists M' such that $(M\phi_e\tau_L)\downarrow = M'\phi_e\tau_L$ and $M\phi_e\tau_R \to^* M'\phi_e\tau_R$. Similarly, there exists N' such that $(N\phi_e\tau_L)\downarrow = N'\phi_e\tau_L$ and $N\phi_e\tau_R \to^* N'\phi_e\tau_R$. From $M'\phi_e\tau_L =_{E'} N'\phi_e\tau_L$ and Claim 2, we deduce $M'\phi_e =_{E'} N'\phi_e$. Therefore $M'\phi_e\tau_R =_{E'} N'\phi_e\tau_R$ and thus $M\phi_e\tau_R =_E N\phi_e\tau_R$, that is $M\phi_e\sigma_{\tilde{N_k}}\tau_R =_E N\phi_e\sigma_{\tilde{N_k}}\tau_R$.

Proof of Claim 1: This result is proved by inspection of the rewrite rules, using the fact that the decryption key sk_T is not deducible. More precisely, assume that $M\phi_e\tau \to U$ for some $\tau \in \{\tau_R, \tau_L\}$. It means that there exists a rewriting rule $l \to r \in \mathcal{R}_E$ and a position p such that $M\phi_e\tau|_p =_{E'} l\theta$ for some θ. p cannot occur below M since $\phi_e\tau$ is in normal form. If $M|_p = l\theta'$ for some θ' then we conclude that we can rewrite M as expected. The only interesting case is thus when $M|_p$ is not an instance of l but $M\phi_e\tau|_p$ is. By inspection of the rules, $l \to r$ can only correspond to one of the three equations E5, E6 or E7. The case of Equations E5 or E6 is ruled out by the fact that sk_T is not deducible from $\phi_e\tau$. The last case is when the rule corresponding to Equation E7 is applied. Then it must be the case that $M|_p = x * y$ with x, y variables of $\mathrm{dom}(\phi_e)$. By construction of ϕ_e, we have that $(x * y)\phi_e \to z\phi_e$ (applying the rule corresponding to Equation E7), thus the result.

Proof of Claim 2: Assume by contradiction that there exist M, N two terms such that $M\phi_e\tau =_{E'} N\phi_e\tau$ for some $\tau \in \{\tau_R, \tau_L\}$ and $M\phi_e \neq_{E'} N\phi_e$. Consider M, N two minimal terms that satisfy this property. By case inspection, it must be the case that M and N are both variables. Thus we have $x\phi_e\tau =_{E'} y\phi_e\tau$ and $x\phi_e \neq_{E'} y\phi_e$ with $x, y \in \mathrm{dom}(\phi_e)$, $x \neq y$. The head symbol of $x\phi_e\tau$ must be penc. Then by construction of ϕ_e, τ does not change the randomness used in penc and the randomness uniquely determines the variable, which implies $x = y$, contradiction. □

We now demonstrate that tallying valid ballots yields the same result in both worlds.

Lemma 9. *Let ℓ, be a number of candidates. Let $N_3, \ldots, N_k$ be terms, free w.r.t. $sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}$. Let $\theta_{\tilde{N_k}} = \{{}^{N_k}/_{y_k} \mid k \in \{3, \ldots, n\}\}$ such that $N_i\theta_{\tilde{N_k}}\sigma\tau$ is a valid ballot for any $\tau \in \{\tau_L, \tau_R\}$. Let $\sigma = \{{}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}, {}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}\}$. Then*

$$result_i\ \theta_{\tilde{N_k}}\sigma\tau_L =_E result_i\ \theta_{\tilde{N_k}}\sigma\tau_R$$

and both $result_j\ \theta_{\tilde{N_k}}\sigma\tau_L$ and $result_j\ \theta_{\tilde{N_k}}\sigma\tau_R$ are terms built from constants one *and* zero *by application of the function symbol $+$.*

Proof. We first define $N'_i = N_i\theta_{\tilde{N_k}}$. By Lemma 7, we know that $\pi_j(N'_i\sigma\tau_L) =_E \mathsf{penc}(z_{\mathsf{pk}}, U^i_j, V^i_j)\sigma\tau_L$ for some free terms U^i_j, V^i_j. By Lemma 8, we know that $\phi_1\tau_L \approx_s \phi_1\tau_R$ thus we can deduce $\pi_j(N'_i\sigma\tau_R) =_E \mathsf{penc}(z_{\mathsf{pk}}, U^i_j, V^i_j)\sigma\tau_R$. The equational theory ensure that $\mathsf{penc}(K, U, V) =_E \mathsf{penc}(K', U', V')$ implies $K =_E K'$, $U =_E U'$, and $V =_E V'$. Thus we deduce $V^i_j\sigma\tau_L =_E V^i_j\sigma\tau_R$. Therefore, we get that $result_j\ \theta_{\tilde{N_k}}\sigma\tau_L =_E x^{\mathsf{vote}}_{1,j}\tau_L + x^{\mathsf{vote}}_{2,j}\tau_L + +(V^3_j + \cdots + V^k_j)\sigma\tau_L =_E x^{\mathsf{vote}}_{1,j}\tau_R + x^{\mathsf{vote}}_{2,j}\tau_R + +(V^3_j + \cdots + V^k_j)\sigma\tau_R =_E result_j\ \theta_{\tilde{N_k}}\sigma\tau_R$.

Moreover, $V^i_j\sigma\tau \in \{\mathsf{one}, \mathsf{zero}\}$ is ensured by the fact that $N_i\theta_{\tilde{N_k}}\sigma\tau$ is a valid ballot. Therefore we deduce that both $result_j\ \theta_{\tilde{N_k}}\sigma\tau_L$ and $result_j\ \theta_{\tilde{N_k}}\sigma\tau_R$ are terms built from constants one and zero by application of the function symbol $+$. □

We finally show that the encrypted ballots of honest voters and the partial decryptions do not leak any information to the adversary.

Lemma 10. *Let $\phi_6 = \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\ .\ (\{{}^{ballot_1}/_{x_1}\} \mid \{{}^{ballot_2}/_{x_2}\} \mid \{{}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\} \mid \{{}^{\mathsf{partial}_j}/_{x^{\mathsf{partial}}_j} \mid 1 \le j \le \ell\})$. We have $\phi_6\sigma_{\tilde{N_k}}\tau_L \approx_s \phi_6\sigma_{\tilde{N_k}}\tau_R$.*

The proof is very similar to the proof of Lemma 8

Proof. First, we decompose ϕ_6 and consider $\phi = \nu\,\tilde{n}.\theta$ where $\tilde{n} = \{sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\}$ and $\theta = \{{}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\} \mid \Big\{\{{}^{\mathsf{partial}_j}/_{x_{\mathsf{partial}_j}}\} \mid \{{}^{ciph_{i,j}}/_{x_{ciph_{i,j}}}\} \mid \{{}^{spk_{i,j}}/_{x_{spk_{i,j}}}\} \mid \{{}^{\widehat{spk_i}}/_{x_{\widehat{spk_i}}}\} \;\Big|\; i \in \{1,2\} \wedge 1 \le j \le \ell\Big\}$. It follows immediately that $\phi_6\tau_L \approx_s \phi_6\tau_R$ if and only if $\phi\tau_L \approx_s \phi\tau_R$.

Secondly, witness that the adversary can arbitrarily combine ciphertexts from the frame – namely, ciphertexts $ciph_{1,1}, ciph_{2,1}, \ldots, ciph_{1,\ell}, ciph_{2,\ell}$ – with ciphertexts in the frame or freshly constructed ciphertexts, we enrich the frame ϕ with any such combination of ciphertexts. Formally, for any $\alpha_j, \beta_j \in \mathbb{N}$ and terms P, R we define $C_{\alpha_1,\ldots,\alpha_\ell,\beta_1,\ldots,\beta_\ell,\alpha_4,P,R}$ as follows:

$$\mathsf{penc}(\mathsf{pk}(sk_T), R \circ \textstyle\bigcirc_{1\le j\le\ell} r^{\alpha_j}_{1,j} \circ r^{\beta_j}_{2,j}, P + \sum_{1\le j\le\ell} \alpha_j.x^{\mathsf{vote}}_{1,j} + \beta_j.x^{\mathsf{vote}}_{2,j})$$

We define the extended frame ϕ_e below.

$$\phi_e = \nu\, \tilde{n}.(\theta \mid \{{}^{C_{\alpha_1,\alpha_2,\alpha_3,\alpha_4,P,R}}/_{x_{\alpha_1,\alpha_2,\alpha_3,\alpha_4,P,R}} \mid \alpha_1,\alpha_2,\alpha_3,\alpha_4 \in \mathbb{N} \text{ and terms } P,R \text{ s.t. } (\mathrm{fn}(P)\cup\mathrm{fn}(R))\cap\tilde{n}=\emptyset, \mathrm{fv}(P,R)\subseteq \mathrm{dom}(\phi_e) \text{ with no cycle}\})$$

Note that ϕ_e is infinite. By Lemma 4, it is sufficient to show $\phi_e\tau_L \approx_s \phi_e\tau_R$. Let $\phi'_e = \phi_e\sigma_{\tilde{N}_k}$. We introduce the following two claims.

Claim 3. *Let M be a term such that $\mathrm{fv}(M) \cap (\mathrm{fv}(\phi'_e) \setminus \mathrm{dom}(\phi'_e)) = \emptyset$ and $\mathrm{fn}(M)\cap\tilde{n}=\emptyset$. If $M\phi'_e\tau \to U$ for some $\tau \in \{\tau_R,\tau_L\}$ then there exists N such that $U =_{E'} N\phi'_e\tau$ and $M\phi'_e\tau' \to N\phi'_e\tau'$ for any $\tau' \in \{\tau_R,\tau_L\}$.*

Claim 4. *Let M,N be two terms such that $(\mathrm{fv}(M)\cup\mathrm{fv}(N)) \cap (\mathrm{fv}(\phi'_e) \setminus \mathrm{dom}(\phi'_e)) = \emptyset$ and $\mathrm{fn}(M,N)\cap\tilde{n}=\emptyset$. If $M\phi'_e\tau =_{E'} N\phi'_e\tau$ for some $\tau \in \{\tau_R,\tau_L\}$ then $M\phi'_e =_{E'} N\phi'_e$.*

The above claims allow the construction of our proof. Let M,N be two terms such that $\mathrm{fn}(M,N)\cap\tilde{n}=\emptyset$ and $M\phi'_e\sigma_{\tilde{N}_k}\tau_L =_E N\phi'_e\sigma_{\tilde{N}_k}\tau_L$. We assume (possibly by renaming) that $(\mathrm{fv}(M)\cup\mathrm{fv}(N))\cap(\mathrm{fv}(\phi'_e)\backslash\mathrm{dom}(\phi'_e))=\emptyset$. We have $M\phi'_e\tau_L =_E N\phi'_e\tau_L$. Thus $(M\phi'_e\tau_L)\downarrow =_{E'} (N\phi'_e\tau_L)\downarrow$. Applying repeatedly Claim 3, we deduce that there exists M' such that $(M\phi'_e\tau_L)\downarrow = M'\phi'_e\tau_L$ and $M\phi'_e\tau_R \to^* M'\phi'_e\tau_R$. Similarly, there exists N' such that $(N\phi'_e\tau_L)\downarrow = N'\phi'_e\tau_L$ and $N\phi'_e\tau_R \to^* N'\phi'_e\tau_R$. From $M'\phi'_e\tau_L =_{E'} N'\phi'_e\tau_L$ and Claim 4, we deduce $M'\phi'_e =_{E'} N'\phi'_e$. Therefore $M'\phi'_e\tau_R =_{E'} N'\phi'_e\tau_R$ and thus $M\phi'_e\tau_R =_E N\phi'_e\tau_R$, that is $M\phi'_e\sigma_{\tilde{N}_k}\tau_R =_E N\phi'_e\sigma_{\tilde{N}_k}\tau_R$.

Proof of Claim 3: This result is proved by inspection of the rewrite rules, using the fact that the decryption key sk_T is not deducible. More precisely, assume that $M\phi'_e\tau \to U$ for some $\tau \in \{\tau_R,\tau_L\}$. It means that there exists a rewriting rule $l \to r \in \mathcal{R}_E$ and a position p such that $M\phi'_e\tau|_p =_{E'} l\theta$ for some θ. p cannot occur below M since $\phi'_e\tau$ is in normal form. If $M|_p = l\theta'$ for some θ' then we conclude that we can rewrite M as expected. The only interesting case is thus when $M|_p$ is not an instance of l but $M\phi'_e\tau|_p$ is. By inspection of the rules, $l \to r$ can only correspond to one of the three equations E5, E6 or E7. The case of Equations E5 is ruled out by the fact that sk_T is not deducible from $\phi'_e\tau$. For Equation E6, it must be the case that $M\phi'_e|_p = result_j\sigma_{\tilde{N}_k}$. Using Lemma 9, we deduce that $M\phi'_e|_p\tau \to R$ modulo E' where R is a sum of ones and zero. Therefore $M\phi'_e\tau \to M[R]_p\phi'_e\tau$. The last case is when the rule corresponding to Equation E7 is applied. Then it must be the case that $M|_p = x * y$ with x,y variables of $\mathrm{dom}(\phi'_e)$. By construction of ϕ'_e, we have that $(x*y)\phi'_e \to z\phi'_e$ (applying the rule corresponding to Equation E7), thus the result.

Proof of Claim 4: Assume by contradiction that there exist M,N two terms such that $M\phi'_e\tau =_{E'} N\phi'_e\tau$ for some $\tau \in \{\tau_R,\tau_L\}$ and $M\phi'_e \neq_{E'} N\phi'_e$. Consider M,N two minimal terms that satisfy this property. By case inspection, it must be the case that M and N are both variables. Thus we have $x\phi'_e\tau =_{E'} y\phi'_e\tau$ and $x\phi'_e \neq_{E'} y\phi'_e$ with $x,y \in \mathrm{dom}(\phi'_e)$, $x \neq y$. The head symbol of $x\phi'_e\tau$

Figure 6 Partial evolutions of the Helios process specification

We introduce some partial evolutions of the Helios process specification:

$$
\begin{array}{lcl}
A^1 & = & \nu\, sk_T, a_2, d, r_{1,1}, \ldots, r_{1,\ell}, y_1 \,.\, (\_ \mid \{^{ballot_1}/_{y_1}\} \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}) \\
A^2 & = & A^1[\_ \mid \{^{ballot_1}/_{x_1}\}] \\
A^3 & = & \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}, y_1, y_2 \,.\, (\_ \mid \\
 & & \quad \{^{ballot_1}/_{y_1}\} \mid \{^{ballot_2}/_{y_2}\} \mid \{^{ballot_1}/_{x_1}\} \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}) \\
A^4 & = & A^3[\_ \mid \{^{ballot_2}/_{x_2}\}] \\
A^5 & = & A^4[\nu\, y_{\mathsf{partial}} . (\_ \mid \{^{partials}/_{y_{\mathsf{partial}}}\}] \\
A^6 & = & A^5[\_ \mid \{^{partials}/_{x_{\mathsf{partial}}}\})] \\
A^7 & = & A^6[\{^{results}/_{x_{\mathsf{result}}}\}]
\end{array}
$$

$$
\begin{array}{lcl}
BB^1_n & = & \overline{c}\langle y_1 \rangle \,.\, BB^2_n \\
BB^2_n & = & a_2(y_2) \,.\, BB^3_n \\
BB^3_n & = & \overline{c}\langle y_2 \rangle \,.\, BB^{'}_{3,n} \\
BB^{'}_{j,n} & = & a_j(y_j) \,.\, \text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{y_j}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
 & & \cdots\ a_n(y_n) \,.\, \text{if } \phi^{\mathsf{sol}}_{\ell,n-1}\{^{y_n}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
 & & BB^4_n \\
BB^{''}_{j,n} & = & \text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{y_j}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
 & & a_{j+1}(y_{j+1}) \,.\, \text{if } \phi^{\mathsf{sol}}_{\ell,j}\{^{y_{j+1}}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
 & & \cdots\ a_n(y_n) \,.\, \text{if } \phi^{\mathsf{sol}}_{\ell,n-1}\{^{y_n}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
 & & BB^4_n \\
BB^4_n & = & \overline{d}\langle (\!| tally_1, \ldots, tally_\ell |\!) \rangle \,.\, BB^5_n \\
BB^5_n & = & d(y_{\mathsf{partial}}) \,.\, BB^6_n \\
BB^6_n & = & \overline{c}\langle y_{\mathsf{partial}} \rangle \,.\, BB^7_n \\
BB^7_n & = & \overline{c}\langle (\!| \mathsf{dec}(\pi_1(y_{\mathsf{partial}}), tally_\ell), \ldots, \mathsf{dec}(\pi_\ell(y_{\mathsf{partial}}), tally_\ell) |\!) \rangle
\end{array}
$$

$$
\begin{array}{lcl}
T^1_\ell & = & \overline{d}\langle partials \rangle
\end{array}
$$

where $partials = (\!| \mathsf{partial}(sk_T, tally_1), \ldots, \mathsf{partial}(sk_T, tally_\ell) |\!)$ and $results = (\!| \mathsf{dec}(\mathsf{partial}(sk_T, tally_1), tally_1), \ldots, \mathsf{dec}(\mathsf{partial}(sk_T, tally_\ell), tally_\ell) |\!)$.

must be penc or partial. Assume first that the head symbol of $x\phi'_e\tau$ is penc. Then by construction of ϕ'_e, τ does not change the randomness used in penc and the randomness uniquely determines the variable, which implies $x = y$, contradiction. Assume now that the head symbol of $x\phi'_e\tau$ is partial. Then it must be the case that $tally_{j_1}\sigma_{\tilde{N}_k}\tau =_{E'} tally_{j_2}\sigma_{\tilde{N}_k}\tau$ while $tally_{j_1}\sigma_{\tilde{N}_k} \neq_{E'} tally_{j_2}\sigma_{\tilde{N}_k}$. This would require $x_{ciph_{i,j_1}}\tau = x'_{ciph_{i',j_2}}\tau$ for some i, i', which is excluded due to the randomness. □

B.4 Proof of Theorem 1

We introduce some partial evolutions of the Helios process specification in Figure 6 and define a relation $\mathcal{R}$ between processes in Figure 7. We clearly have that

Figure 7 Definition of the relation $\mathcal{R}$

Consider the smallest relation $\mathcal{R}$ which is closed under structural equivalence and includes the following pairs of extended processes, where for all $3 \leq j \leq n$, terms M, terms $N_1, \ldots, N_j$, substitutions $\sigma = \{{}^{N_k}/_{y_k} \mid k \in \{3, \ldots, n\}\}$ and distinct variables $x_{\mathsf{partial}}, x_{\mathsf{result}}, x_1, x_2$ such that $N_j\sigma\tau_L$ and $N_j\sigma\tau_R$ are valid ballot, $\mathrm{fv}(M) \cup \bigcup_{3 \leq i \leq j} \mathrm{fv}(N_i) \subseteq \mathrm{dom}(A^4)$ and $(\mathrm{fn}(M) \cup \bigcup_{3 \leq i \leq j} \mathrm{fn}(N_i)) \cap \mathrm{bn}(A^4) = \emptyset$.

$$A^{\phi^{\mathsf{sol}}}_{\ell,n}[V\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma'], \quad A^{\phi^{\mathsf{sol}}}_{\ell,n}[V\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma' \mid V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma] \tag{R1}$$

$$A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^1_n \mid T_\ell]\tau_L, \quad A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB^1_n \mid T_\ell]\tau_R \tag{R2}$$

$$A^2[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^2_n \mid T_\ell]\tau_L, \quad A^2[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB^2_n \mid T_\ell]\tau_R \tag{R3}$$

$$A^3[BB^3_n \mid T_\ell]\tau_L, \quad A^3[BB^3_n \mid T_\ell]\tau_R \tag{R4}$$

$$A^4[BB^{'}_{j,n}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_L, \quad A^4[BB^{'}_{j,n}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_R \tag{R5}$$

$$A^4[BB^{''}_{j,n}\{{}^{N_k}/_{y_k} \mid k \in \{3, \ldots, j-1\}\}\{{}^{M}/_{y_j}\} \mid T_\ell]\tau_L, \quad A^4[BB^{''}_{j,n}\{{}^{N_k}/_{y_k} \mid k \in \{3, \ldots, j-1\}\}\{{}^{M}/_{y_j}\} \mid T_\ell]\tau_R \tag{R6}$$

$$A^4[0 \mid T_\ell]\tau_L, \quad A^4[0 \mid T_\ell]\tau_R \tag{R7}$$

$$A^4[BB^4_n\tau \mid T_\ell]\tau_L, \quad A^4[BB^4_n\tau \mid T_\ell]\tau_R \tag{R8}$$

$$A^4[BB^5_n \mid T^1_\ell]\tau\tau_L, \quad A^4[BB^5_n \mid T^1_\ell]\tau\tau_R \tag{R9}$$

$$A^5[BB^6_n]\tau\tau_L, \quad A^5[BB^6_n]\tau\tau_R \tag{R10}$$

$$A^6[BB^7_n]\tau\tau_L, \quad A^6[BB^7_n]\tau\tau_R \tag{R11}$$

$$A^7\tau\tau_L, \quad A^7\tau\tau_R \tag{R12}$$

$A^{\phi}_{\ell,n}[V\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma'] \; \mathcal{R} \; A^{\phi}_{\ell,n}[V\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma' \mid V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$. We now wish to show that $\mathcal{R} \cup \mathcal{R}^{-1}$ satisfies the three properties of Definition 2. By symmetry we focus on $\mathcal{R}$. Overwriting the definition, we may say that a term N is a valid ballot if both $N\sigma\tau_L$ and $N\sigma\tau_R$ are valid ballots, where σ is defined

Figure 7.

Static equivalence. We must show for all extended processes A and B, where $A \mathcal{R} B$, that $A \approx_s B$. By Lemma 4, it is sufficient to show $A^7\tau\tau_L \approx_s A^7\tau\tau_R$ for any $N_3, \ldots, N_n$ valid ballots. Let $\phi_7 = \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}\,.\, (\{^{ballot_1}/_{x_1}\} \mid \{^{ballot_2}/_{x_2}\} \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\} \mid \{^{(\!|partial_1, \ldots, partial_n|\!)}/_{x_{\mathsf{partial}}}\} \mid \{^{(\!|result_1, \ldots, result_n|\!)}/_{x_{\mathsf{result}}}\})$. We have to show $\phi_7\sigma_{N_3,\ldots,N_n}\tau_L \approx_s \phi_7\sigma_{N_3,\ldots,N_n}\tau_R$. By Lemma 9, we deduce that $(\!|result_1, \ldots, result_\ell|\!)\sigma_{N_3,\ldots,N_n}\tau_L = (\!|result_1, \ldots, result_\ell|\!)\sigma_{N_3,\ldots,N_n}\tau_R$ and is equal to a constant (always deducible) term. Thus by Lemma 3, it is sufficient to show that $\phi_6\sigma_{N_3,\ldots,N_n}\tau_L \approx_s \phi_6\sigma_{N_3,\ldots,N_n}\tau_R$, where ϕ_6 as defined in Lemma 10. We conclude by Lemma 10.

Internal reductions. We must show for all extended processes A and B, where $A \mathcal{R} B$, that if $A \to A'$ for some A', then $B \to^* B'$ and $A' \mathcal{R} B'$ for some B'. We observe that if $A \equiv A^1[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^1_n \mid T_\ell]\tau_L$ and $B \equiv A^1[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB^1_n \mid T_\ell]\tau_R$ – that is, $A \mathcal{R} B$ by (R2) – then there is no extended process A' such that $A \to A'$; similarly, for (R4), (R5), (R7), (R10), (R11) and (R12). We proceed by case analysis on the remaining cases.

(R1) We have $A \equiv A^{\phi^{\mathsf{sol}}}_{\ell,n}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma']$ and $B \equiv A^{\phi^{\mathsf{sol}}}_{\ell,n}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma' \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$. If $A \to A'$, then it must be the case that $A \equiv C[\overline{a_1}\langle y_1\rangle.0 \mid a_1(y_1).BB^1_n]\tau_L$ and $A' \equiv C[0 \mid BB^1_n]\tau_L$, where $C[\_] = A^1[\nu\, a_1.(\_ \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid T_\ell)]$. It follows from $B \equiv C'[\overline{a_1}\langle y_1\rangle.0 \mid a_1(y_1).BB^1_n]\tau_R$, that $B \to B'$, where $C'[\_] = A^1[\nu\, a_1.(\_ \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid T_\ell)]$ and $B' = A^1[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB^1_n \mid T_\ell]\tau_R$. Since $A^1[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^1_n \mid T_\ell]\tau_L \mathcal{R} B'$ and $A' \equiv A^1[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^1_n \mid T_\ell]\tau_L$, we derive $A' \mathcal{R} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

(R3) This case is similar to (R1). We have $A \equiv A^2[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB^2_n \mid T_\ell]\tau_L$ and $B \equiv A^2[V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB^2_n \mid T_\ell]\tau_R$. If $A \to A'$, then it must be the case that $A \equiv C[\overline{a_2}\langle y_2\rangle.0 \mid a_2(y_2).BB^3_n]\tau_L$ and $A' \equiv C[0 \mid BB^3_n]\tau_L$, where $C[\_] = A^3[\nu\, a_2.(\_ \mid T_\ell)]$. It follows from $B \equiv C[\overline{a_2}\langle y_2\rangle.0 \mid a_2(y_2).BB^3_n]\tau_R$, that $B \to B'$, where $B' = A^3[BB^3_n \mid T_\ell]\tau_R$. Since $A^3[BB^3_n \mid T_\ell]\tau_L \mathcal{R} B'$ and $A' \equiv A^3[BB^3_n \mid T_\ell]\tau_L$, we derive $A' \mathcal{R} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

(R6) We have $A \equiv A^4[BB^{''}_{j,n}\{^{N_k}/_{y_k} \mid k \in \{3, \ldots, j-1\}\}\{^{M}/_{y_j}\} \mid T_\ell]\tau_L$ and $B \equiv A^4[BB^{''}_{j,n}\{^{N_k}/_{y_k} \mid k \in \{3, \ldots, j-1\}\}\{^{M}/_{y_j}\} \mid T_\ell]\tau_R$ for some integer $j \in \{3, \ldots, n\}$, valid ballots $N_3, \ldots, N_{j-1}$ and term M such that $\mathrm{fv}(M) \cup \bigcup_{3 \le i \le j-1} \mathrm{fv}(N_i) \subseteq \mathrm{dom}(A^4)$ and $(\mathrm{fn}(M) \cup \bigcup_{3 \le i \le j-1} \mathrm{fn}(N_i)) \cap \mathrm{bn}(A^4) = \emptyset$.

If $A \to A'$, then it must be the case that $A \equiv C[\mathrm{if}\ \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\}\ \mathrm{then}\ P\ \mathrm{else}\ 0]\tau_L$, where $C[\_] = A^4[\_ \mid T_\ell]$. Furthermore, if $j < n$, then $P = BB^{'}_{j+1,n}\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\}\{^{M}/_{y_j}\}$; otherwise $P = BB^4_n\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots,$

$j-1\}\}\{^{M}/_{y_j}\}$. We also have $B \equiv C[\text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\} \text{ then } P \text{ else } 0]\tau_R$.

Let $\sigma_R = \{{}^{ballot_1\tau_R}/_{x_1}, {}^{ballot_2\tau_R}/_{x_2}, {}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}$, we have $ballot_1\tau_R$ is syntactically equal to $x_1\sigma_R$ and $ballot_2\tau_R$ is syntactically equal to $x_2\sigma_R$, it follows that $B \equiv C[\text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1\sigma_R}/_{y_1}, {}^{x_2\sigma_R}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\} \text{ then } P \text{ else } 0]\tau_R$ and, moreover, since $\varphi(C[0]\tau_R) = \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}, y_1, y_2.\sigma_R$ we have $B \equiv C[\text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\} \text{ then } P \text{ else } 0]\tau_R$. We proceed by case analysis on the structure of A':

- If $A' \equiv C[P]\tau_L$, then by closure of internal reduction under structural equivalence we have $C[\text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\} \text{ then } P \text{ else } 0]\tau_L \rightarrow C[P]\tau_L$ because $ballot_1\tau_L$ is syntactically equal to $x_1\sigma_L$, $ballot_2\tau_L$ is syntactically equal to $x_2\sigma_L$ and $\varphi(C[0]\tau_L) = \nu\, sk_T, d, r_{1,1}, \ldots, r_{1,\ell}, r_{2,1}, \ldots, r_{2,\ell}, y_1, y_2.\sigma_L$, where $\sigma_L = \{{}^{ballot_1\tau_L}/_{x_1}, {}^{ballot_2\tau_L}/_{x_2}, {}^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}$.

  Assume A and B satisfy the preconditions of Corollary 1, it follows that $B \rightarrow B' = A^4[P \mid T_\ell]\tau_R$. We now prove our assumption. Since $A \mathcal{R} B$, it follows by Condition 1 of Definition 2 that $A \approx_s B$. Let $\phi = \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\}$. By inspection of $\phi^{\mathsf{sol}}_{\ell,j-1}$, we have $\mathrm{fn}(\phi) = \mathrm{fn}(M) \cup \bigcup_{3\leq i\leq j-1} \mathrm{fn}(N_i)$ and since $\mathrm{bn}(C) = \mathrm{bn}(A^4)$ it follows that $\mathrm{bn}(C) \cap \mathrm{fn}(\phi) = \emptyset$; we also have $\mathrm{fv}(\phi) = \{x_1, x_2, z_{\mathsf{pk}}\} \cup \mathrm{fv}(M) \cup \bigcup_{3\leq i\leq j-1} \mathrm{fv}(N_i)$ and since $\mathrm{dom}(C) = \{x_1, x_2, z_{\mathsf{pk}}\}$ it follows that $\mathrm{fv}(\phi) \subset \mathrm{dom}(C)$. We have shown that the preconditions of Corollary 1 are satisfied, hence $B \rightarrow B' = A^4[P \mid T_\ell]\tau_R$. It remains to show $A' \mathcal{R} B'$.

  We know $[\![\phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\}\sigma_L]\!] = \mathsf{true}$ and $[\![\phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\}\sigma_R]\!] = \mathsf{true}$; it follows, for $\tau \in \{\tau_L, \tau_R\}$, that $[\![\phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}\}\{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}, {}^{ballot_1}/_{y_1}, {}^{ballot_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\}\tau]\!] = \mathsf{true}$ and we know that M is a valid ballot. We continue by case analysis on the structure of P:

  1. If $P = BB^{'}_{j+1,n}\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\}\{^{M}/_{y_j}\}$, then we have $j < n$. Let $j' = j+1$ and $N_j = M$, observe $P = BB^{'}_{j',n}\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j\}\}$ and $A^4[BB^{'}_{j',n}\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j\}\} \mid T_\ell]\tau_L \mathcal{R} B'$. The result $A' \mathcal{R} B'$ follows by closure of $\mathcal{R}$ under structural equivalence.
  2. If $P = BB^4_n\{^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\}\{^{M}/_{y_j}\}$, then it must be the case that $j = n$. Let $N_j = M$ and hence $P = BB^4_n\tau$. Since $A^4[BB^4_n\tau \mid T_\ell]\tau_L \mathcal{R} B'$ and $A' \equiv A^4[BB^4_n\tau \mid T_\ell]\tau_L$, we derive $A' \mathcal{R} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

- $A' \equiv C[0]\tau_L$, then similarly to above we have $C[\text{if } \phi^{\mathsf{sol}}_{\ell,j-1}\{^{M}/_{y_{\mathsf{ballot}}}, {}^{x_1}/_{y_1}, {}^{x_2}/_{y_2}, {}^{N_3}/_{y_3}, \ldots, {}^{N_{j-1}}/_{y_{j-1}}\} \text{ then } P \text{ else } 0]\tau_L \rightarrow C[0]\tau_L$ and it follows by Corollary 1 that $B \rightarrow B' = C[0]\tau_R$. Since $A^4[0 \mid T_\ell] \mathcal{R} B'$

and $A' \equiv A^4[0 \mid T_\ell]$, we derive $A' \mathrel{\mathcal{R}} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

(R8) We have $A \equiv A^4[BB_n^4\tau \mid T_\ell]\tau_L$ and $B \equiv A^4[BB_n^4\tau \mid T_\ell]\tau_R$, where $BB_n^4 = \overline{d}\langle(\!|tally_1, \ldots, tally_\ell|\!)\rangle \,.\, BB_n^5$ and $T_\ell = d(y_{\mathsf{tally}}) \,.\, \overline{d}\langle(\!|\mathsf{partial}(sk_T, \pi_1(y_{\mathsf{tally}})), \ldots, \mathsf{partial}(sk_T, \pi_\ell(y_{\mathsf{tally}}))|\!)\rangle$. If $A \to A'$, then it must be the case that $A' \equiv A^4[BB_n^5 \mid T_\ell^1]\tau\tau_L$. It follows immediately that $B \to B'$, where $B' = A^4[BB_n^5 \mid T_\ell^1]\tau\tau_R$. We derive $A' \mathrel{\mathcal{R}} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

(R9) We have $A \equiv A^4[BB_n^5 \mid T_\ell^1]\tau\tau_L$ and $B \equiv A^4[BB_n^5 \mid T_\ell^1]\tau\tau_R$. If $A \to A'$, then it must be the case that $A \equiv A^5[\overline{d}\langle y_{\mathsf{partial}}\rangle.0 \mid d(y_{\mathsf{partial}}).BB_n^6\}]\tau\tau_L$ and $A' \equiv A^5[0 \mid BB_n^6]\tau\tau_L$. It follows from $B \equiv A^5[d(y_{\mathsf{partial}}).BB_n^6 \mid \overline{d}\langle y_{\mathsf{partial}}\rangle.0]\tau\tau_R$ that $B \to B'$, where $B' = A^5[BB_n^6]\tau\tau_R$. We derive $A' \mathrel{\mathcal{R}} B'$ by the closure of $\mathcal{R}$ under structural equivalence.

Labelled reductions. We must show for all extended processes A and B, where $A \mathrel{\mathcal{R}} B$, that if $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'. We observe cases (R1), (R3), (R6), (R7), (R8), (R9) and (R12) cannot be reduced by labelled reductions and proceed by case analysis on the remaining cases.

(R2) We have $A \equiv A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB_n^1 \mid T_\ell]\tau_L$ and $B \equiv A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB_n^1 \mid T_\ell]\tau_R$. If $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then it must be the case that $A \equiv A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid \overline{c}\langle y_1\rangle.BB_n^2 \mid T_\ell]\tau_L$ and $A' \equiv A^2[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB_n^2 \mid T_\ell]\tau_L$ for some variable x_1 where $\alpha = \nu\, x_1.\overline{c}\langle x_1\rangle$ and $x_1 \neq z_{\mathsf{pk}}$. It follows from $B \equiv A^1[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid \overline{c}\langle y_1\rangle.BB_n^2 \mid T_\ell]\tau_R$, that $B \xrightarrow{\alpha} B'$ where $B' = A^2[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma \mid BB_n^2 \mid T_\ell]\tau_R$. We have $A^2[V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\sigma' \mid BB_n^2 \mid T_\ell]\tau_L \mathrel{\mathcal{R}} B'$ and by closure of $\mathcal{R}$ under structural equivalence $A' \mathrel{\mathcal{R}} B'$.

(R4) We have $A \equiv A^3[BB_n^3 \mid T_\ell]\tau_L$ and $B \equiv A^3[BB_n^3 \mid T_\ell]\tau_R$. If $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then it must be the case that $A \equiv A^3[\overline{c}\langle y_2\rangle.BB_{3,n}^{'} \mid T_\ell]\tau_L$ and $A' \equiv A^4[BB_{3,n}^{'} \mid T_\ell]\tau_L$ for some variable x_2, where $\alpha = \nu\, x_2.\overline{c}\langle x_2\rangle$ and $x_2 \notin \{x_1, z_{\mathsf{pk}}\}$. It follows from $B \equiv A^3[\overline{c}\langle y_2\rangle.BB_{3,n}^{'} \mid T_\ell]\tau_R$, that $B \xrightarrow{\alpha} B'$, where $B' = A^4[BB_{3,n}^{'} \mid T_\ell]\tau_R$. Since $A^4[BB_{3,n}^{'} \mid T_\ell]\tau_L = A^4[BB_{j,n}^{'}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_L$ and $A^4[BB_{3,n}^{'} \mid T_\ell]\tau_R = A^4[BB_{j,n}^{'}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_R$ when $j = 3$, we have $A^4[BB_{3,n}^{'} \mid T_\ell]\tau_L \mathrel{\mathcal{R}} B'$ and derive $A' \mathrel{\mathcal{R}} B'$ by closure of $\mathcal{R}$ under structural equivalence $A' \mathrel{\mathcal{R}} B'$.

(R5) We have $A \equiv A^4[BB_{j,n}^{'}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_L$ and $B \equiv A^4[BB_{j,n}^{'}\{{}^{N_k}/_{y_k} \mid j > 3 \wedge k \in \{3, \ldots, j-1\}\} \mid T_\ell]\tau_R$ for some integer $j \in \{3, \ldots, n\}$ and terms $N_3, \ldots, N_{j-1}$, where $\bigcup_{3 \leq i \leq j-1} \mathrm{fv}(N_i) \subseteq \mathrm{dom}(A^4)$ and $\mathrm{bn}(A^4) \cap \bigcup_{3 \leq i \leq j-1} \mathrm{fn}(N_i) = \emptyset$. If $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then it must be the case that

$A \equiv A^4[a_j(y_j).BB''_{j,n}\{N_k/y_k \mid j > 3 \wedge k \in \{3,\ldots,j-1\}\} \mid T_\ell]\tau_L$ and $A' \equiv A^4[BB''_{j,n}\{N_k/y_k \mid j > 3 \wedge k \in \{3,\ldots,j-1\}\}\{M/y_j\} \mid T_\ell]\tau_L$, where $\alpha = c(M)$ for some term M. It follows from $B \equiv A^4[a_j(y_j).BB''_{j,n}\{N_k/y_k \mid j > 3 \wedge k \in \{3,\ldots,j-1\}\} \mid T_\ell]\tau_R$, that $B \xrightarrow{\alpha} B'$, where $B' = A^4[BB''_{j,n}\{N_k/y_k \mid j > 3 \wedge k \in \{3,\ldots,j-1\}\}\{M/y_j\} \mid T_\ell]\tau_R$. We have $A^4[BB''_{j,n}\{N_k/y_k \mid k \in \{3,\ldots,j-1\}\}\{M/y_j\} \mid T_\ell]\tau_L \mathcal{R}\ B'$, and derive $A' \mathcal{R}\ B'$ by closure of $\mathcal{R}$ under structural equivalence.

(R10) We have $A \equiv A^5[BB^6_n]\tau\tau_L$ and $B \equiv A^5[BB^6_n]\tau\tau_R$. If $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then it must be the case that $A' \equiv A^6[BB^7_n]\tau\tau_L$ for some variable x_{partial}, where $\alpha = \nu\, x_{\mathsf{partial}}.\overline{c}\langle x_{\mathsf{partial}}\rangle$ and $x_{\mathsf{partial}} \notin \{x_1, x_2, z_{\mathsf{pk}}\}$. It follows immediately that $B \xrightarrow{\alpha} B'$, where $B' = A^6[BB^7_n]\tau\tau_R$. We have $A^6[BB^7_n]\tau\tau_L \mathcal{R}\ B'$ and by closure of $\mathcal{R}$ under structural equivalence $A' \mathcal{R}\ B'$.

(R11) This case is similar to (R10). We have $A \equiv A^6[BB^7_n]\tau\tau_L$ and $B \equiv A^6[BB^7_n]\tau\tau_R$. If $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then it must be the case that $A' \equiv A^7\tau\tau_L$ for some variable x_{result}, where $\alpha = \nu\, x_{\mathsf{result}}.\overline{c}\langle x_{\mathsf{result}}\rangle$ and $x_{\mathsf{result}} \notin \{x_1, x_2, x_{\mathsf{partial}}, z_{\mathsf{pk}}\}$. It follows immediately that $B \xrightarrow{\alpha} B'$, where $B' = A^7\tau\tau_R$. We have $A^7\tau\tau_L \mathcal{R}\ B'$ and by closure of $\mathcal{R}$ under structural equivalence $A' \mathcal{R}\ B'$.

References

[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL'01: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 104–115. ACM Press, 2001.

[2] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In *IFIP TCS'00: 1st International Conference on Theoretical Computer Science*, volume 1872 of *LNCS*, pages 3–22. Springer, 2000.

[3] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

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

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

[6] Ben Adida. Attacks and Defenses. Helios documentation, `http://documentation.heliosvoting.org/attacks-and-defenses`, 2010.

[7] 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.

[8] Ben Adida and Olivier Pereira. Private email communication, November 2010.

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

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

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

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

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

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

[15] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot secrecy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, LNCS. Springer, 2011. To appear.

[16] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, February–March 2008.

[17] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *EUROCRYPT'87: 4th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 304 of *LNCS*, pages 127–141. Springer, 1988.

[18] 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.

[19] 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.

[20] 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.

[21] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *FOCS'85: 26th Foundations of Computer Science Symposium*, pages 383–395. IEEE Computer Society, 1985.

[22] Benny Chor and Michael O. Rabin. Achieving Independence in Logarithmic Number of Rounds. In *PODC'87: 6th Principles of Distributed Computing Symposium*, pages 260–268. ACM Press, 1987.

[23] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. Technical Report 2007-2081, Cornell University, May 2007. Revised March 2008.

[24] 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.

[25] Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. Cryptology ePrint Archive, Report 2010/625, 2010.

[26] 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.

[27] 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.

[28] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Autority 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.

[29] 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.

[30] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *CSFW'06: 19th Computer Security Foundations Workshop*, pages 28–42. IEEE Computer Society, 2006.

[31] 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.

[32] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster. 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 289–309. Springer, 2010.

[33] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[34] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography. In *STOC'91: 23rd Theory of computing Symposium*, pages 542–552. ACM Press, 1991.

[35] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *Journal on Computing*, 30(2):391–437, 2000.

[36] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Vote-Independence: A Powerful Privacy Notion for Voting Protocols. In *FPS'11: 4th Workshop on Foundations & Practice of Security*, LNCS. Springer, 2011. To appear.

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

[38] Est Républicain. June, 18th 2007. Meurthe-et-Moselle edition (Daily French Newspaper).

[39] Saghar Estehghari and Yvo Desmedt. Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example. In *EVT/WOTE'10: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2010.

[40] *Article L65 of the French electoral code.* `http://www.legifrance.gouv.fr/`.

[41] *Résultat par bureau du premier tour des élections régionales*, 2010. `http://www.monaulnay.com/wp-content/uploads/2010/03/resultat_regionale_par_bureau.pdf`.

[42] 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.

[43] Rosario Gennaro. Achieving independence efficiently and securely. In *PODC'95: 14th Principles of Distributed Computing Symposium*, pages 130–136. ACM Press, 1995.

[44] Rosario Gennaro. A Protocol to Achieve Independence in Constant Rounds. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):636–647, 2000.

[45] 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`, May 2010.

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

[47] 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.rsa.com/rsalabs/node.asp?id=2860`.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] Lucie Langer. *Privacy and Verifiability in Electronic Voting*. PhD thesis, Fachbereich Informatik, Technischen Universität Darmstadt, 2010.

[53] Lucie Langer, Axel Schmidt, Johannes Buchmann, and Melanie Volkamer. A Taxonomy Refining the Security Requirements for Electronic Voting: Analyzing Helios as a Proof of Concept. In *ARES'10: 5th International Conference on Availability, Reliability and Security*, pages 475–480. IEEE Computer Society, 2010.

[54] Lucie Langer, Axel Schmidt, Johannes Buchmann, Melanie Volkamer, and Alexander Stolfik. Towards a Framework on the Security Requirements for Electronic Voting Protocols. In *Re-Vote'09: First International Workshop on Requirements Engineering for E-Voting Systems*, pages 61–68. IEEE Computer Society, 2010.

[55] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In *ICISC'03: 6th International Conference on Information Security and Cryptology*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.

[56] Arjen K. Lenstra and Hendrik W. Lenstra Jr. Algorithms in Number Theory. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 12, pages 673–716. MIT Press, 1990.

[57] Adrian Leung, Liqun Chen, and Chris J. Mitchell. On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). In *Trust'08: 1st International Conference on Trusted Computing and Trust in Information Technologies*, number 4968 in LNCS, pages 179–190. Springer, 2008.

[58] Jia Liu. A Proof of Coincidence of Labeled Bisimilarity and Observational Equivalence in Applied Pi Calculus. `http://lcs.ios.ac.cn/~jliu/papers/LiuJia0608.pdf`, 2011.

[59] Tatsuaki Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In *SP'97: 5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 25–35. Springer, 1998.

[60] Miriam Paiola. Extending ProVerif's Resolution Algorithm for Verifying Group Protocols. Master's thesis, Faculty of Mathematical, Physical and Natural Science, University of Padova, 2010.

[61] Miriam Paiola and Bruno Blanchet. Automatic Verification of Group Protocols with Unbounded Numbers of Participants and Sessions. Unpublished draft, 2011.

[62] Participants of the Dagstuhl Conference on Frontiers of E-Voting. *Dagstuhl Accord*, 2007. `http://www.dagstuhlaccord.org/`.

[63] Torben P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *EUROCRYPT'91: 10th International Conference on the Theory and Applications of Cryptographic Techniques*, number 547 in LNCS, pages 522–526. Springer, 1991.

[64] Olivier Pereira, Ben Adida, and Olivier de Marneffe. Bringing open audit elections into practice: Real world uses of helios. Swiss e-voting workshop, `https://www.e-voting-cc.ch/images/sevot10/slides/helios_20100906.pdf`. See also `http://www.uclouvain.be/crypto/electionmonitor/`, 2010.

[65] Birgit Pfitzmann. Breaking Efficient Anonymous Channel. In *EUROCRYPT'94: 11th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 950 of *LNCS*, pages 332–340. Springer, 1994.

[66] Princeton University. *Princeton Election Server*, 2010. `https://princeton-helios.appspot.com/`.

[67] Carsten Rudolph. Covert Identity Information in Direct Anonymous Attestation (DAA). In *SEC'07: 22nd International Information Security Conference*, volume 232 of *International Federation for Information Processing (IFIP)*, pages 443–448. Springer, 2007.

[68] Mark D. Ryan and Ben Smyth. Applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011.

[69] Peter Y. A. Ryan and Steve A. Schneider. An Attack on a Recursive Authentication Protocol. A Cautionary Tale. *Information Processing Letters*, 65(1):7–10, 1998.

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

[71] 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.

[72] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 148–164. Springer, 1999.

[73] Berry Schoenmakers. Voting Schemes. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook, Second Edition, Volume 2: Special Topics and Techniques*, chapter 15. CRC Press, 2009.

[74] Daniel Shanks. Class number, a theory of factorization and genera. In *Number Theory Institute*, volume 20 of *Symposia in Pure Mathematics*, pages 415–440. American Mathematical Society, 1971.

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

[76] Ben Smyth and Véronique Cortier. Attacking ballot secrecy in Helios. YouTube video, linked from `http://www.bensmyth.com/publications/10-attacking-helios/`, 2010.

[77] Ben Smyth and Véronique Cortier. Replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, June 2011. `http://hal.inria.fr/inria-00599182/`.

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

[79] Melanie Volkamer. *Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities*, volume 30 of *Lecture Notes in Business Information Processing*. Springer, 2009.

[80] Melanie Volkamer and Rüdiger Grimm. Determine the Resilience of Evaluated Internet Voting Systems. In *Re-Vote'09: First International Workshop on Requirements Engineering for E-Voting Systems*, pages 47–54. IEEE Computer Society, 2010.

[81] Bogdan Warinschi. A Computational Analysis of the Needham-Schröeder-(Lowe) Protocol. In *CSFW'03: 16th Computer Security Foundations Workshop*, pages 248–262. IEEE Computer Society, 2003.

[82] Bogdan Warinschi. A computational analysis of the Needham-Schroeder-(Lowe) protocol. *Journal of Computer Security*, 13(3):565–591, 2005.

[83] Douglas Wikström. Simplified Submission of Inputs to Protocols. Cryptology ePrint Archive, Report 2006/259, 2006.

[84] Douglas Wikström. Simplified Submission of Inputs to Protocols. In *SCN'08: 6th International Conference on Security and Cryptography for Networks*, volume 5229 of *LNCS*, pages 293–308. Springer, 2008.

Adapting Helios for provable ballot privacy

David Bernhard[1], Véronique Cortier[2], Olivier Pereira[3],
Ben Smyth[2], Bogdan Warinschi[1]

[1] University of Bristol, England
[2] LORIA - CNRS, France
[3] Université Catholique de Louvain, Belgium

Abstract. Recent results show that the current implementation of Helios, a practical e-voting protocol, does not ensure independence of the cast votes, and demonstrate the impact of this lack of independence on vote privacy. Some simple fixes seem to be available and security of the revised scheme has been studied with respect to symbolic models.
In this paper we study the security of Helios using computational models. Our first contribution is a model for the property known as ballot privacy that generalizes and extends several existing ones.
Using this model, we investigate an abstract voting scheme (of which the revised Helios is an instantiation) built from an arbitrary encryption scheme with certain functional properties. We prove, generically, that whenever this encryption scheme falls in the class of *voting-friendly* schemes that we define, the resulting voting scheme provably satisfies ballot privacy.
We explain how our general result yields cryptographic security guarantees for the revised version of Helios (albeit from non-standard assumptions).
Furthermore, we show (by giving two distinct constructions) that it is possible to construct voting-friendly encryption, and therefore voting schemes, using only standard cryptographic tools. We detail an instantiation based on ElGamal encryption and Fiat-Shamir non-interactive zero-knowledge proofs that closely resembles Helios and which provably satisfies ballot privacy.

1 Introduction

Electronic voting protocols have the potential to offer efficient and sound tallying with the added convenience of remote voting. It is therefore not surprising that their use has started to gain ground in practice: USA, Norway and Estonia are examples of countries where e-voting protocols have been, at the very least, trialled in elections on a national scale.

Due to the sensitive nature of elections, security of e-voting protocols is crucial and has been investigated extensively. Among the security properties that have been identified for e-voting, perhaps the most desirable one is that users' votes should remain confidential. Three levels of confidentiality have been identified. These are (in increasing strength) the following.

– Ballot privacy: A voter's vote is not revealed to anyone.
– Receipt–freeness: A voter cannot obtain information which can prove to a coercer how she voted.
– Coercion resistance: Even a voter who collaborates with a coercer cannot obtain information that proves how she voted.

Other important properties that are desirable include ballot independence [12] (the ballots cast do not depend on each other) and end-to-end verifiability [23, 28, 38] (it is possible to verify that the election process has been followed honestly).

This paper is motivated by recent developments regarding the security of the Helios voting scheme [45]. Starting from version 2.0 [35], Helios has been using a variant of a classical protocol by Cramer et al. [14] incorporating tweaks proposed by Benaloh [29], and has been used in real-world elections, for example by the International Association for Cryptographic Research (IACR) to elect its 2010 board [36], by Princeton University to elect the undergraduate student government [46] and to elect the president of the Université Catholique de Louvain [35]. Helios aims to achieve only ballot privacy and explicitly discards the stronger confidentiality notions (which it does not satisfy) in favor of efficiency. It turns out that the current implementation of Helios does *not* enforce ballot independence (contrary to the original protocol of Cramer et al. [14]) and, as a result, Cortier and Smyth [37, 42] have exhibited several attacks against the ballot privacy property of Helios. (The property is called "ballot secrecy" in Cortier and Smyth's papers.) The attacks range from simple ballot copying to subtle reuse of parts of existing ballots, however they can all be detected (and prevented) by public algorithms. A revised scheme has been proved secure in a symbolic model but its security in the stronger, computational sense has not been assessed.

Contributions. We start by providing a *computational* security model for ballot privacy (Section 2). In a sense, our model generalizes and strengthens the model of [24, 26] where an attacker tries to distinguish when two ballots are swapped. Here, we ask that the adversary cannot detect whether the ballots cast are ballots for votes that the adversary has chosen or not. In doing so, the adversary is allowed to control arbitrarily many players and see the result of the election. Our model uses cryptographic games and thus avoids imposing the more onerous constraints that other definitional styles (in particular simulability) require from protocols.

Next we turn our attention to the revised version of Helios. Our analysis follows a somewhat indirect route: instead of directly analysing the scheme as it has been implemented, we analyze an abstract version of Helios that follows the same basic architecture, but where the concrete primitives are replaced with more abstract versions. Of course, the version we analyze implements the suggested weeding of ballots. We present this abstract scheme as a generic construction of a voting scheme starting from encryption scheme with specific functional and security properties (Section 3).

Focusing on this more abstract version brings important benefits. Firstly, we pin-down more clearly the requirements that the underlying primitives should

satisfy. Specifically, we identify a class of *voting-friendly* encryption schemes which when plugged in our construction yield voting schemes with provable ballot privacy. Roughly speaking, such encryption schemes are IND-CCA2 secure and have what we call a homomorphic embedding (parts of the ciphertexts can be seen as ciphertexts of a homomorphic encryption scheme). Secondly, our analysis applies to all voting schemes obtained as instantiations of our generic construction. Although we analyze and propose constructions which for efficiency reasons resort to random oracles, our generic approach also invites other (non-random oracle based) instantiations.

Next, we show how to construct voting-friendly encryption schemes using standard cryptographic tools (Section 4). We discuss two distinct designs. The first construction starts from an arbitrary (IND-CPA) homomorphic encryption scheme and attaches to its ciphertexts a zero-knowledge proof of knowledge of the plaintext. We refer to this construction as the Enc+PoK construction. Despite its intuitive appeal, we currently do not know how to prove that the above design leads to an IND-CCA2 secure encryption scheme (a proprety demanded by voting-friendliness). We therefore cannot conclude the security of our generic scheme when implemented with an arbitrary Enc+PoK scheme. Nevertheless, an investigation into this construction is important since the instantiation where Enc is the ElGamal scheme and PoK is obtained using the Fiat-Shamir paradigm applied to a Schnorr-like protocol corresponds precisely to the encryption scheme currently used in Helios. The security of this specific construction has been analyzed in prior work. Tsiounis and Yung [17] and Schnorr and Jakobsson [19] demonstrate that the scheme is IND-CCA2 secure, but their proofs rely on highly non-standard assumptions. Nevertheless, in conjunction with the security of our main construction, one can conclude that the current implementation of Helios satisfies ballot privacy based on either the assumption in [17] or those of [19].

We then take a closer look at the Enc+PoK construction and revisit a technical reason that prevents an IND-CCA2 security proof, first studied by Shoup and Gennaro [16]. Very roughly, the problem is that the knowledge extractor associated to the proof of knowledge may fail if used multiple times since its associated security guarantees are only for constant (or logarithmically many) uses. With this in mind, we note that a security proof is possible if the proof of knowledge has a so called *straight line* extractor [22]. This type of extractor can be reused polynomially many times. In this case, the Enc+PoK construction leads to a voting-friendly encryption scheme, whenever Enc is an arbitrary IND-CPA homomorphic encryption scheme.

The second design uses the well-known Naor-Yung transformation [7]. We show that if the starting scheme is an arbitrary (IND-CPA) homomorphic encryption scheme then the result of applying the NY transform is a voting-friendly encryption scheme. Applied generically, the transform may lead to non-efficient schemes (one of its components is a simulation-sound zero-knowledge proof of membership [18]). We present a related construction (where the proof of membership is replaced by a proof of knowledge) which can be efficiently instantiated in the random oracle model. In the final section of the paper (Section 5) we pro-

pose adopting an instantiation of Helios where the encryption-friendly scheme is implemented as above. The computational overhead for this scheme is reasonable (and can be further improved through specific optimization) and the scheme comes with the formal guarantees offered by the results of this paper.

Related work. Chevallier-Mames *et al.* [27] present an unconditional definition of ballot privacy but Helios cannot be expected to satisfy this definition due to its reliance on computational assumptions. Chevallier-Mames additionally show that their definition of unconditional ballot privacy is incompatible with universal verifiability; however, ballot privacy and universal verifiability have been shown to coexist under weaker assumptions, for example as witnessed by Juels, Catalano & Jakobsson [23]. Computational definitions of ballot privacy have been considered by Benaloh *et al.* [2, 4, 5]. These definitions however do not come with a general characterization of the properties that an encryption scheme should satisfy in order to ensure that they are satisfied (the corresponding security notions did not exist at that time either). Wikström [34] considered the general problem of secure submission of inputs with applications to mixnet-based voting protocols. His definitions and constructions are the most closely related to ours, and will be discussed below. Other definitions for voting systems have been proposed in terms of UC realization of ideal voting functionalities, starting with Groth [21], which capture privacy as part of the functionality behavior.

In addition, receipt-freeness has been considered by Benaloh & Tuinstra [11] and Moran & Naor [25] and coercion resistance has been studied by Juels, Catalano & Jakobsson [23], Küsters, Truderung & Vogt [40] and Unruh & Müller-Quade [39]. These definitions can be used to show ballot privacy because it is believed to be a weaker condition [11, 26]; however, they are too strong for protocols which only provide ballot privacy and in particular, they cannot be used to analyse ballot privacy in Helios. Ballot privacy has also been formalized in the symbolic model (for example, [26, 33]) but the symbolic model suffers a serious weakness: In general, a correct security proof does not imply the security of the protocol. Cortier & Smyth [37, 42] present an attack against ballot privacy in Helios and propose a variant of Helios which aims to prevent the attack by weeding ballots. Their solution has been shown to satisfy ballot privacy in the symbolic model but Cortier & Smyth acknowledge that a thorough cryptographic analysis of the solution is necessary.

2 Ballot privacy

Notation Throughout this paper, we use the following notation. Assignment and input/output of algorithms are both denoted by a left-facing arrow $\leftarrow$. Picking a value x uniformly at random from a set S is denoted by $x \xleftarrow{R} S$. The expression $C \xleftarrow{+} c$ appends c to the list C, () on its own is an empty list. We use "C" style returns in algorithms, i.e. "Return $a = b$" to mean return 1 if $a = b$, otherwise 0. A function f is called *negligible* if for any polynomial P, there exists η_0 such that $\forall \eta \geq \eta_0$, $f(\eta) \leq \frac{1}{P(\eta)}$.

2.1 Voting Schemes

In this section we fix a general syntax for the class of voting schemes that we treat in this paper. In particular, our syntax encompasses several variations of the Helios protocol.

We consider schemes for votes in a non-empty set $\mathbb{V}$, and we assume $\perp$ to be a special symbol not in $\mathbb{V}$ that indicates that the voter has abstained. The result of an election is then an arbitrary function ρ that takes a list of votes as input and returns the election result. Elections are stateful, so the algorithms that we define next use such a state. Since often, and in particular in the case of Helios, this state is a bulletin board, in the definition below we write BB for this state (and even refer to it as a bulletin board).

Definition 1 (Voting scheme). *Algorithms* (Setup, Vote, ProcessBallot, Tally) *define a voting scheme as follows.*

Setup *The setup algorithm takes a security parameter* 1^λ *as input and returns secret information* x*, public information* y*, and initializes the state* BB*. We write* $(x, y, BB) \leftarrow \mathsf{Setup}(1^\lambda)$ *for this process. We assume the public information is available to all subsequent algorithms.*

Vote *The voting algorithm takes a vote* $v \in \mathbb{V}$ *as input and produces as output a ballot* b *(that encodes the vote). We write* $b \leftarrow \mathsf{Vote}(v)$ *for this process.*

ProcessBallot *The ballot processing algorithm takes a candidate ballot* b *and a bulletin board* BB*, checks the ballot for correctness (e.g. that it is well formed, it is not a duplicate, etc.) and returns a result (accept/reject) and the new state of the bulletin board. We write* $(a, BB) \leftarrow \mathsf{ProcessBallot}(BB, b)$ *for this process. Here* a *is either* accept *or* reject.

Tally *The tallying algorithm takes the secret information* x *and the bulletin board* BB *and produces the election result.*

For correctness of the scheme, we demand two conditions: 1) ballot tallying corresponds to evaluating the function ρ on the underlying votes; and 2) correctly constructed votes will be accepted by the ballot processing algorithm. Both conditions should hold with overwhelming probability and can be captured by the experiment described in Figure 1. In this experiment, an adversary repeatedly submits votes $v_1, v_2, \ldots \in \mathbb{V}$ and each vote is used to construct a ballot which is then processed. The game outputs 1 (the adversary wins) if the ProcessBallot algorithm rejects some ballot or the result of the election does not correspond to the votes cast. The voting scheme is correct if the algorithm outputs 1 with at most negligible probability.

2.2 Security Model

Informally, ballot privacy is satisfied if an adversary in control of arbitrarily many voters cannot learn anything about the votes of the remaining, honest voters beyond what can be inferred from the election result. The adversary can read the (public) bulletin board and the communication channels between the

$\mathbf{Exp}_{\Pi}^{corr}(A)$
$(x, y, BB) \leftarrow \mathsf{Setup}$
$V = ()$
repeat
$(a, v) \leftarrow A$
$b \leftarrow \mathsf{Vote}(v)$
$(r, BB) \leftarrow \mathsf{ProcessBallot}(BB, b)$
$V \xleftarrow{+} v$
until $a = \mathsf{stop}$ or $r = \mathsf{reject}$
if $r =$"reject" or $\mathsf{Tally}(x, BB) \neq \rho(V)$ then return 1 else return 0

Fig. 1. Experiment for defining the correctness of a voting scheme.

honest parties and the bulletin board (in other words, we assume them to be authentic but not secret). Ballot privacy requires that the adversary is unable to distinguish between real ballots and fake ballots, where ballots are replaced by ballots for some fixed vote ε chosen by the adversary.

Formally, we consider an adversary that can issue two types of queries, vote and ballot, to an oracle $\mathcal{O}$. The oracle maintains two bulletin boards initialized via the setup algorithm: BB is visible to the adversary and BB' always contains ballots for the real votes. A vote query causes a ballot for the given vote to be placed on the hidden BB'. In the real world, the same ballot is placed on BB; in the fake one a ballot for ε is placed on BB instead. A ballot query always causes the submitted ballot to be processed on both boards. This process is defined formally in Figure 2. The experiment on the right of Figure 2 is used to define ballot privacy. The selection of β corresponds to the real world ($\beta = 0$) or the fake world ($\beta = 1$). Throughout the experiment the adversary has access to BB, but tallying is done using BB'.

Definition 2 (Ballot Privacy). *We define the advantage of adversary A in defeating ballot privacy for voting scheme Π by:*

$$\mathbf{Adv}_{\Pi}^{BS}(A) = \Pr[\mathbf{Exp}_{\Pi}^{BS}(A) = 1] - \frac{1}{2}$$

and say that Π ensures ballot privacy if for any efficient adversary its advantage is negligible.

We make a few remarks regarding the security model that we propose. Firstly, we use cryptographic games rather than a simulation based definition. The former offer well-accepted levels of security, are more flexible, and allow for more efficient implementations. Second, we model directly the more relaxed notion of vote privacy and not stronger notions like receipt-freeness or coercion resistance [26]. While stronger notions are certainly desirable, they are more difficult to achieve leading to rather inefficient protocols. Indeed, Helios deliberately trades these stronger notions for efficiency. Finally, we emphasize that our computational definition does not mirror existing security definitions in more

$\mathsf{vote}(v)$
$b' \leftarrow \mathsf{Vote}(v)$
if $\beta = 0$ then $b \leftarrow b'$
else $b \leftarrow \mathsf{Vote}(\varepsilon)$
$(r, BB) \leftarrow \mathsf{ProcessBallot}(b, BB)$
$(r', BB') \leftarrow \mathsf{ProcessBallot}(b', BB')$
return (r, BB, b)

$\mathsf{ballot}(b)$
$(r, BB) \leftarrow \mathsf{ProcessBallot}(b, BB)$
if $r = \mathsf{accept}$ then
$(r', BB') \leftarrow \mathsf{ProcessBallot}(b, BB')$
return (r, BB)

$\mathbf{Exp}_{\Pi}^{BS}(A)$
$(x, y, BB) \leftarrow \mathsf{Setup}(1^{\lambda})$
$BB' \leftarrow BB$
$(\varepsilon, st) \leftarrow A(y)$
$\beta \leftarrow \{0, 1\}$
$st \leftarrow A^{\mathcal{O}}(st)$
$\mathsf{result} \leftarrow \mathsf{Tally}(x, BB')$
$\hat{\beta} \leftarrow A(st, \mathsf{result})$
return $\beta = \hat{\beta}$

Fig. 2. The algorithms on the left explain how the oracle processes adversary's queries. The experiment on the right is used to define ballot privacy.

abstract models, e.g. [24]. It turns out that the direct extension of that definition to computational models seems strictly weaker than the definition that we provide. We comment more on this point later in the paper.

3 A generic construction of voting schemes with ballot privacy

In this section we present a generic construction of a voting scheme starting from any encryption scheme with certain properties. We first fix this class of encryption schemes (which we call voting-friendly), then give our construction and prove its security.

3.1 Voting-Friendly Encryption

In a nutshell, a voting-friendly encryption scheme is a "(threshold) checkable provable IND-CCA2 secure public key encryption scheme with key derivation and a homomorphic embedding". These rather convoluted looking requirements are in fact not too onerous. We explain informally each of the requirements in turn and give formal definitions. For simplicity, the presentation in this section is for the non-threshold case, that is decryption is carried out using a single key by a single party, as opposed to implementing decryption via an interactive process where several parties share the keys.

Non-Interactive Zero Knowledge Proof Systems. Here we recall some basic notions regarding non-interactive zero-knowledge proof systems [6]. Given language L_R defined by NP relation R we write $(w, x) \in R$ if w is the witness that $x \in L_R$. A proof system for L_R is given by a pair of algorithms $(\mathsf{Prover}, \mathsf{Verifier})$ called

prover and verifier, respectively. We distinguish between proof systems in the common reference string model (in this situation, an additional algorithm Setup produces a common reference string accessible to both the prover and the verifier) and the random oracle model (where the setup is not required, but all algorithms in the system have access to a random oracle). In a standard execution of the proof system, the prover and the verifier both have an element $x \in L_R$ as input and in addition, the prover has as input a witness w that $x \in L_R$ (i.e. $R(w, x) = 1$). The prover sends a single message π to the verifier who outputs the decision to accept/reject. We call π a proof for the statement $x \in L_R$. Typical requirements for such proof systems are that they should be sound (if the input x is not in L_R then the verifier rejects π with overwhelming probability) and complete (if x is in the language then the verifier accepts π with probability 1). We write $\pi \leftarrow \mathsf{Prover}$ for the process of producing proof π when the statement x and the witness w are clear from the context. A non-interactive proof system is zero-knowledge if there exists a simulator Sim that is able to produce transcripts indistinguishable from those of a normal execution of the protocol. The simulator may use a trapdoor in the common reference string model, or can program the random oracle in the random oracle model. We occasionally write $(\mathsf{Prover}, \mathsf{Verifier}) : R$ to indicate that the proof system is for the language L_R.

We assume the reader is familiar with public key encryption and its associated security notions. We write $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for the key generation, encryption, and decryption algorithms of a public key encryption scheme.

Homomorphic encryption. We also briefly recall the notion of *homomorphic* encryption. An encryption scheme is homomorphic if the plaintext space is a group and there exists an algorithm Add that takes two ciphertexts for messages m_0 and m_1 and produces a ciphertext for $m_0 \circ m_1$ (where $\circ$ is the group operation on plaintexts).

Embeddable Encryption. A crucial property for the encryption schemes that are the focus of this section is that they have a *homomorphic embedding*. Informally, this property means that it is possible to identify part(s) of the ciphertexts as forming a ciphertext for some other encryption scheme, and this second encryption scheme is homomorphic. The ElGamal+PoK construction sketched in the previous section is an example of an encryption scheme with an homomorphic embedding. Indeed the e component of a ciphertext (e, π) is a ciphertext for an homomorphic encryption scheme (ElGamal). The next definition makes this discussion more precise.

Definition 3 (Homomorphic Embedding). *We say that the homomorphic encryption scheme* $\Pi = (\mathsf{EGen}, \mathsf{EEnc}, \mathsf{EDec}, \mathsf{EAdd})$ *is embedded in encryption scheme* $\Pi' = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, or alternatively that encryption scheme* Π' *has* Π *as a* homomorphic embedding *if there are algorithms* $\mathsf{ExtractKey}, \mathsf{Extract}$ *such that for all* m, pk, sk, c

$$\mathsf{EGen}() = \mathsf{ExtractKey}(\mathsf{Gen}())$$

$$\mathsf{EEnc}(m, \mathsf{ExtractKey}(pk)) = \mathsf{Extract}(\mathsf{Enc}(m, pk))$$

$$\mathsf{Dec}(c, sk) = \mathsf{EDec}(\mathsf{Extract}(c), sk)$$

Essentially, the ExtractKey algorithm maps keys (or key pairs) for the "larger" scheme to keys for the embedded one, and the Extract algorithm extracts the ciphertext for the embedded scheme out of ciphertext for the larger one, while performing validity verifications at the same time.

The Extract algorithm must, by definition, produce a ciphertext that decrypts to the same value as the input that it is given; in particular it must produce a "ciphertext" that decrypts to $\perp$ if and only if its input does. However, the Extract algorithm does not take any secret keys as input. This implies that anyone can check whether a ciphertext is valid (in the sense that it decrypts to something other than $\perp$) without knowing the secret key. This property forms the basis for combining homomorphic and IND-CCA2 secure encryption in our construction.

We note that an IND-CCA2 secure cryptosystem with homomorphic embedding is actually very close to a submission secure augmented (SSA) cryptosystem as defined by Wikström [34]. Some important differences appear, though. The most important one is that SSA cryptosystems do not require public verifiability of the ciphertexts: it might be necessary to publish a private key augmentation to be able to perform ciphertext validity checks. While this feature enables efficient solutions that are secure in the standard model, it is however often not desirable in practice: it is quite useful to be able to dismiss invalid votes as soon as they are submitted (and to resolve potential conflicts at that time) rather than needing to wait for some partial key to be revealed. Besides, in order to mitigate this inconvenience, SSA cryptosystems allow multiple independent augmentations, which enables updating an augmentation and revealing the previous one in order to be able to check the validity of previously submitted ciphertexts. Our requirement of immediate public verifiability property makes this feature unnecessary for our purpose.

We also note that in concurrent work, Persiano [44] and Smart [41] define similar embedding concepts.

S2P Key Derivation. This property simply requires that if a key pair is produced by the key generation algorithm of an encryption scheme then it is possible to compute the public key from the secret key. This property will allow us to use proofs of knowledge of the secret key corresponding to the public key.

Definition 4 (S2P Key Derivation). *An encryption scheme has the* S2P key derivation *property if there is an algorithm* DeriveKey *such that* $(x, y) \leftarrow \mathsf{Gen}$ *implies* $y = \mathsf{DeriveKey}(x)$.

Provable Encryption. In our generic construction voters need to certify that various encryptions in the ballots that they produce satisfy some desirable properties (e.g. that a ciphertext encrypts 0 or 1, and not something else), and such certification can be done via zero-knowledge proofs of knowledge. Since all of the

statements that we are interested in are NP statements, the existence of appropriate proof systems follows from general results [9]. Here, we make more precise the statements for which we demand the existence of such proof systems and introduce some useful notation for the proof systems associated to the various languages that we define.

In particular, it should be possible to prove knowledge of the secret key corresponding to the public key, knowledge of the plaintext underlying a ciphertext, as well as proving that a certain plaintext has been obtained by decrypting with the key associated to the public key. To avoid complex nomenclature, we call a scheme for which this is possible a scheme with provable encryption.

Definition 5 (Provable Encryption). *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* provable *if it has the S2P key derivation property and the following non-interactive zero-knowledge proof systems exist:*

1. $(\mathsf{ProveGen}, \mathsf{VerifyGen})$: $R_1(x, y) := y \stackrel{?}{=} \mathsf{DeriveKey}(x)$
2. $(\mathsf{ProveEnc}, \mathsf{VerifyEnc})$: $R_2((m, r), c) := c \stackrel{?}{=} \mathsf{Enc}(m; r)$
3. $(\mathsf{ProveDec}, \mathsf{VerifyDec})$: $R_3(x, (c, y, d)) := y \stackrel{?}{=} \mathsf{DeriveKey}(x) \wedge d \stackrel{?}{=} \mathsf{Dec}(x, c)$

The above definition is for standard encryption schemes. For the case when the encryption scheme that we need is embedded, we demand in addition the existence of proof systems for the following two properties. The first requires that one can prove a statement that involves plaintexts underlying several ciphertexts, and secondly, one should be able to prove that the keys for the embedded schemes in use have been correctly obtained from the keys of the embedding one. This latter condition is a simple adaptation of provability as defined above.

Definition 6 (Provable Embedding). *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for message space* M *with embedded scheme* $(\mathsf{EGen}, \mathsf{EEnc}, \mathsf{EDec})$ *has* embedded provability *for* $M' \subseteq M^N$ *(for some* $N \in \mathbb{N}$*) if the following zero-knowledge proof-systems exist:*

1. $(\mathsf{ProveGen}, \mathsf{VerifyGen})$: $R_4(x, y) := y \stackrel{?}{=} \mathsf{DeriveKey}(x)$
2. $(\mathsf{ProveEnc}, \mathsf{VerifyEnc})$: $R_5((m_1, m_2, \ldots, m_N, r_1, r_2, \ldots, r_N), (c_1, c_2, \ldots, c_N)) :=$

$$\bigwedge_{i=1}^{N} c_i \stackrel{?}{=} \mathsf{Enc}(m_i; r_i) \wedge (m_1, \ldots, m_N) \in M'$$

3. $(\mathsf{ProveEDec}, \mathsf{VerifyEDec})$: $R_6(x, (y, d, c)) :=$

$$y \stackrel{?}{=} \mathsf{DeriveKey}(x) \wedge (x', y') \leftarrow \mathsf{ExtractKey}(x, y) \wedge d \stackrel{?}{=} \mathsf{EDec}(x', c)$$

In the last relation, the second conjunct is not a boolean condition, but simply indicates that the keypair (x', y') is derived from (x, y) using the $\mathsf{ExtractKey}$ algorithm.

The following definition states all the properties that we require from an encryption scheme in order to be able to implement our generic voting scheme.

Definition 7 (Voting-Friendly Encryption). *A voting-friendly encryption scheme for vote space $\mathbb{V}$ is a public-key scheme for message space M with $\mathbb{V} \subseteq M^N$ such that it is IND-CCA2 secure and has S2P key derivation, an embedded homomorphic scheme and embedded provability for $\mathbb{V}$.*

Note that voting-friendly encryption requires security guarantees of both the encryption scheme and the contained proof systems.

3.2 Our Generic Construction

In this section we describe a voting scheme based on an arbitrary voting-friendly encryption scheme. The design idea is similar to that of Helios.

The scheme handles elections with multiple candidates. In an election with three candidates a vote is a triple (a, b, c) such that $a, b, c \in \{0, 1\}$ and $a+b+c = 1$. A ballot is then simply formed by individually encrypting each component of the list with an IND-CCA scheme that has an homomorphic embedding, and proving in zero-knowledge that the individual plaintexts in a ballot satisfy the desired relation. To prevent an adversary from casting a vote somehow related to that of an honest voter, we ensure that each ballot cast does not contain any ciphertexts that are duplicates of ones in the ballots already on the bulletin board. This condition is checked while processing ballots.

More formally, denote the set of ciphertexts contained in a ballot b by $\mathsf{Cipher}(b)$ and the set of all ciphertexts on the bulletin board BB by $\mathsf{Cipher}(BB)$, that is $\mathsf{Cipher}(BB) = \bigcup_{b' \in BB} \mathsf{Cipher}(b')$. When submitting a ballot b, we check that $\mathsf{Cipher}(b) \cap \mathsf{Cipher}(BB) = \emptyset$.

Definition 8 (Abstract Voting Scheme). *Let Π be a voting-friendly encryption scheme. The abstract voting scheme $V(\Pi)$ is the construction consisting of algorithms 1–4.*

In our construction, $\mathcal{V}$ is the set of voters, $\mathcal{Z}$ is a party representing "the public" (elements sent to $\mathcal{Z}$ are published) which also functions as a trusted party for generating the initial setup parameters and $\mathcal{T}$ is the trustee of the election (that receives the decryption keys).

If M is the message space of the voting-friendly encryption scheme we consider the space of votes to be $\mathbb{V} \subseteq M^N$ for some $N \in \mathbb{N}$.

We consider result functions of the form $\rho : \mathbb{V}^* \to M^*$ where $\mathbb{V}^* := \cup_{i \in \mathbb{N}_0} \mathbb{V}^i$ (this allows us to tally an arbitrary number of votes) and each component of the range of ρ can be described by a sum of the form $\rho_k = \sum_{i \in \mathbb{N}} a_{i,k} \cdot v_i$ for constants $a_{i,k} \in \mathbb{N}$. This covers the class of result functions that can be computed homomorphically, including normal and weighted sums of votes but also the special case of revealing all the votes and allows us to exploit the homomorphism in the tallying operation: The same operation can be performed on homomorphic ciphertexts using the EAdd algorithm, for which we write $\oplus$ i.e. $a \oplus b := \mathsf{EAdd}(a, b)$. Furthermore, we can define scalar multiplication $\otimes$ on the ciphertexts i.e. $2 \otimes a := \mathsf{EAdd}(a, a)$.

Algorithm 1 $\mathsf{Setup}(1^\lambda)$

$\mathcal{Z}$:

$params \leftarrow \mathsf{Setup}(1^\lambda)$. These parameters are implicitly available to all further algorithms.

$BB \leftarrow ()$

$\mathcal{T}$:

$(x, y) \leftarrow \mathsf{Gen}(1^\lambda)$

$\pi^{Gen} \leftarrow \mathsf{ProveGen}(x, y)$

$\mathcal{Z} \leftarrow (y, \pi^{Gen})$

$\mathcal{Z}$:

$\mathsf{VerifyGen}(y, \pi^{Gen}) \stackrel{?}{=} 1$ or abort with failure.

Algorithm 2 $\mathsf{Vote}((v_1, v_2, \ldots, v_N))$

$\forall j \in \{1, 2, \ldots, N\}$

$c_j \leftarrow \mathsf{Enc}(y, v_j)$

$\pi_j^b \leftarrow \mathsf{ProveEnc}(y, v_j, c_j)$

$b_j \leftarrow (c_j, \pi_j^b)$

output b

Algorithm 3 $\mathsf{ProcessBallot}(b, BB)$

if $\mathsf{VerifyEnc}(b) = 0$ **then return** ("reject", BB) **end if**

for all $c \in \mathsf{Cipher}(b)$ **do**

if $\mathsf{Extract}(c) = \perp$ **then return** ("reject", BB) **end if**

if $\mathsf{Cipher}(b) \cap \mathsf{Cipher}(BB) \neq \emptyset$ **then return** ("reject", BB) **end if**

end for

$BB \stackrel{+}{\leftarrow} b$

return ("accept", BB)

Algorithm 4 $\mathsf{Tally}(BB)$

for all $c_j \in BB$ $(j \in \mathcal{V})$ **do** $e'_j \leftarrow \mathsf{Extract}(c_j)$ **end for**

for all k **do**

$e''_k \leftarrow \bigoplus_{j \in \mathcal{V}} (a_{j,k} \otimes e'_j)$ {I.e. use EAdd to compute ciphertexts for the results.}

$r_k \leftarrow \mathsf{EDec}(x, e''_k)$

$\pi_k^{Dec} \leftarrow \mathsf{ProveEDec}(x, e''_k, r_k)$

end for

$\mathcal{Z} \leftarrow (r_k, \pi_k^{Dec})_k$

Algorithm 5 Verification

$\mathcal{Z}$ performs the following, aborting if any of the checks (denoted by $\stackrel{?}{=}$) fail. The ordering on $\mathcal{V}$ is a slight abuse of notation; it represents the order the ballots were received in. If successful, the result of the election is r.

$\mathsf{VerifyGen}(y, \pi^{Gen}) \stackrel{?}{=} 1$
for all $j \in \mathcal{V}$ **do**
  $(c_j, \pi_j^b) \leftarrow b_j$
  $\mathsf{VerifyEnc}(b_j) \stackrel{?}{=} 1$
  $(c_j \notin (c_{j'})_{j' \in \mathcal{V}, j' < j}) \stackrel{?}{=} 1$
  $e'_j \leftarrow \mathsf{Extract}(c_j)$
  $e'_j \stackrel{?}{\neq} \perp$
end for
$e' \leftarrow \mathsf{EAdd}(\rho, (e'_j)_{j \in \mathcal{V}})$
$\mathsf{VerifyEDec}(r, \pi^{Dec}, e') \stackrel{?}{=} 1$

We also provide a public verification algorithm as Algorithm 5 although we do not define this property formally.

We only prove ballot privacy of our construction formally; correctness follows from the correctness of the voting-friendly encryption scheme. The following theorem states that ballot privacy relies entirely on the security of the underlying voting-friendly scheme.

Theorem 1. *Let Π be a voting-friendly encryption scheme. Then $V(\Pi)$ has ballot privacy.*

To prove the theorem we proceed in two steps. First, we strip the voting scheme of the unnecessary details that concern verifiability, resulting in a scheme that we call "mini-voting". We prove that ballot privacy for this latter scheme only relies on the IND-CCA2 security of the encryption scheme employed (which highlights IND-CCA2 security as the crucial property needed from the underlying building block). We then explain how to adapt the proof to show the security of $V(\Pi)$.

The full proof can be found in the full version of this paper.

4 Constructions for voting–friendly schemes

In the previous section we gave a generic construction of a voting scheme with ballot privacy starting from an arbitrary voting-friendly encryption scheme. In this section we show that such schemes can be easily constructed using standard cryptographic tools in both the standard and the random oracle models. We discuss three different possibilities.

Encrypt + PoK. This construction does not lead immediately to a voting-friendly scheme but its security is highly relevant to that of Helios, and the design idea forms the basis of a construction that we discuss later.

Under this paradigm, one attempts to construct an IND-CCA2 scheme starting from an IND-CPA scheme and adding to the ciphertext a non-interactive proof of knowledge of the underlying plaintext. Intuitively, this ensures that an adversary cannot make use of a decryption oracle (since he must know the underlying plaintext of any ciphertext) hence the security of the scheme only relies on IND-CPA security. Unfortunately, this intuition fails to lend itself to a rigorous proof, and currently the question whether Enc+PoK yields an IND-CCA2 scheme is widely open. A detailed treatment of the problem first appeared in [16].

Yet, the question is important for the security of Helios: the current implementation is essentially an instantiation of our generic construction with an Enc+PoK encryption scheme. More precisely the encryption scheme Enc is ElGamal, and the proof of knowledge is obtained by applying the Fiat-Shamir transform to a Schnorr proof. Per the above discussion, no general results imply that the resulting ElGamal+PoK scheme is IND-CCA2 secure (a requirement for voting-friendliness) and our generic result does not apply. However, if one is prepared to accept less standard assumptions, two existing results come in handy. The security of the particular construction that employs ElGamal encryption and Fiat-Shamir zero-knowledge proofs of knowledge has been investigated by Tsiounis & Yung [17] and Schnorr & Jakobsson [19]. Both works support the conjecture that the construction is IND-CCA2 but neither result is fully satisfactory. Tsiounis & Yung make a knowledge assumption that essentially sidesteps a crucial part in the security proof, whereas the proof of Schnorr & Jakobsson assumes both generic groups [13] and random oracles [10]. Nevertheless, since using either assumption we can show that ElGamal+PoK construction is a voting-friendly scheme, we conclude that Helios satisfies ballot privacy under the same assumptions. Unfortunately, the security of the construction under standard assumptions is a long-standing open question. This observation motivates the search for alternative constructions of voting-friendly schemes.

Straight-line Extractors. To motivate the construction that we discuss now, it is instructive to explain why a proof that Enc+PoK is IND-CCA2 fails. In such a proof, when reducing the security of the scheme to that of the underlying primitive, a challenger would need to answer the decryption queries of the adversary. Since the underlying encryption scheme is only IND-CPA secure, the only possibility is to use the proof of knowledge to extract the plaintext underlying the queried ciphertexts. Unfortunately here the proof gets stuck. Current definitions and constructions for proofs of knowledge only consider single statements and the knowledge extractor works for polylogarithmically many proofs but it may break down (run in exponential time [19]) for polynomially many. Since the IND-CCA2 adversary is polynomially bounded answering all of its decryption queries may thus not be feasible.

A construction that gets around this problem employs a zero-knowledge proof of knowledge with a *straight-line* extractor. Such extractors do not need to rewind the prover and in this case the Enc+PoK construction yields an IND-CCA2 encryption scheme. This notion of extraction and a variation of the Fiat-

Shamir transform that turns a sigma-protocol into a non-interactive proof of knowledge with a straight-line extractor in the random oracle model has recently been proposed by Fischlin [22]. As above, starting with a homomorphic encryption scheme would yield a voting friendly encryption scheme. Unfortunately the construction in that paper is not suffficiently efficient to yield a practical encryption scheme.

The Naor-Yung Transformation. This transformation starts from any IND-CPA secure encryption scheme. An encryption of message m is simply two distinct encryptions c_1 and c_2 of m under the original scheme, together with a simulation-sound zero-knowledge proof π that c_1 and c_2 encrypt the same message with an extra property that we call unique applicability. Formally, we have the following definition.

Definition 9 (Naor-Yung Transformation). *Let $E = (\mathsf{EGen}, \mathsf{EEnc}, \mathsf{EDec})$ be a public-key encryption system. Let $P = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ be a non-interactive zero-knowledge proof scheme for proving (in Camenisch's notation [15])*

$$\mathbf{PoK}\{(m, r_1, r_2) : \ c_1 = \mathsf{Enc}(y_1, m; r_1) \wedge c_2 = \mathsf{Enc}(y_2, m; r_2)\}$$

with uniquely applicable proofs. Assume the input to Prove *is given in the form $(m, y_1, y_2, r_1, r_2, c_1, c_2)$.*

The Naor-Yung transformation [7] $NY(E, P)$ of the encryption system is the public-key cryptosystem defined in Algorithm 6.

Algorithm 6 Naor-Yung Transformation

Gen
  $(x_1, y_1) \leftarrow \mathsf{EGen}$
  $(x_2, y_2) \leftarrow \mathsf{EGen}$
  return $((x_1, x_2), (y_1, y_2))$
$\mathsf{Enc}((y_1, y_2), m; (r_1, r_2))$
  $c_1 \leftarrow \mathsf{Enc}(y_1, m; r_1)$
  $c_2 \leftarrow \mathsf{Enc}(y_2, m; r_2)$
  $\pi \leftarrow \mathsf{Prove}(m, y_1, y_2, r_1, r_2, c_1, c_2)$
  return (c_1, c_2, π)
$\mathsf{Dec}(c_1, c_2, \pi)$
  if $\mathsf{Verify}(c_1, c_2, \pi) = 1$ **then return** $\mathsf{EDec}(x_1, c_2)$ **else return** $\perp$ **end if**

Sahai [18] showed that the above transformation yields an IND-CCA2 encryption scheme if the starting scheme is IND-CPA and the proof system is simulation-sound and has uniquely applicable proofs (essentially each proof can only be used to prove one statement).

Theorem 2 (Sahai[18]). *If the zero-knowledge proof system P has uniquely applicable proofs then the Naor-Yung transformation $NY(E, P)$ of an IND-CPA secure scheme E gives IND-CCA2 security.*

It turns out that if the starting encryption scheme is homomorphic, then the resulting construction is a voting-friendly encryption scheme. Indeed, the resulting scheme has a homomorphic embeding (given either the first or the second component of the ciphertext) and it is checkable (the checking algorithm only needs to verify the validity of π). As explained earlier, the required proof-systems for provability of the embeding exist, from general results. One can therefore obtain voting schemes with provable ballot privacy in the standard model starting from any homomorphic encryption scheme that is IND-CPA secure in the standard model.

In general, the above construction may not be very efficient (the simulation-sound zero-knowledge proof and associated required proof-systems may be rather heavy). In the random oracle model one can implement the above idea efficiently by replacing the simulation-sound zero-knowledge proof (of membership) with a zero-knowledge proof of knowledge of the message that underlies the two ciphertexts. Interestingly, one may regard the NY transform as providing the underlying encryption scheme with a straight-line extractor (so our previous results already apply).

The following theorem is a variation of the basic Naor-Yung transform applied to our setting.

Theorem 3. *If E is an IND-CPA secure homomorphic encryption scheme with S2P key derivation and P is a zero-knowledge proof of knowledge system with uniquely applicable proofs, then $NY(E, P)$ is a voting friendly encryption scheme.*

5 Application to the Helios protocol

We propose an enhanced version of Helios 3.0 which is an instantiation of our generic voting scheme with a voting-friendly encryption scheme obtained from ElGamal encryption [1] via the NY transform [7]. The required proof of knowledge is obtained via the Fiat-Shamir transform [3] applied to generalized Schnorr proofs. In this scheme duplicate ballots would be rejected as defined in the ProcessBallot procedure (Algorithm 3). We can further improve the efficiency by reusing some components as described by [20].

Thanks to Theorems 1 and 3, we deduce that the enhanced version of Helios 3.0 (provably) preserves ballot privacy. The modification of Helios we propose does not change the architecture nor the trust assumption of Helios and can be easily implemented. The computational overhead is reasonable (both the length of the messages and the time of computation would at most double and some optimizations can be foreseen). In exchange, we get the formal guarantee that Helios does preserve ballot privacy, a very crucial property in the context of electronic voting. For concreteness, we prove the details of the construction, as well as a proof of security in the full version of this paper.

We emphasize that our results go beyond proving ballot privacy of a particular e-voting protocol. We have identified IND-CCA2 as a sufficient condition for constructing voting schemes satisfying our notion of ballot privacy and have

given an abstract construction of a Helios-type voting scheme from IND-CPA secure homomorphic threshold encryption and non-interactive zero-knowledge proofs of knowledge. Our construction is independent of any hardness assumptions or security models (in particular, the random oracle model). We have formalized the concept of embeddable encryption and showed how to construct IND-CCA2 secure encryption with homomorphic embedding, despite the known impossibility of homomorphic IND-CCA2 secure encryption.

As further work, we plan to extend the definitions and proofs for threshold encryption scheme in order to have a fully complete proof for Helios. We are confident that our proof techniques will apply in a straightforward way. We also wish to investigate the possibility of defining ballot privacy in a more general way, e.g. allowing the current voting algorithm to be replaced by a protocol. Indeed, it could the case that casting a vote or tallying the vote require more than one step.

Acknowledgements We are very grateful to Ben Adida for helpful discussions on how to enhance ballot privacy in Helios.

This work was partially supported by the European Commission through the ICT Programme under Contract ICT- 2007-216676 ECRYPT II, by the Interuniversity Attraction Pole P6/26 BCRYPT, and by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 258865 (ProSecure project). Olivier Pereira is a Research Associate of the Belgian Funds for Scientific Research (F.R.S.-FNRS).

References

1. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In: IEEE transactions on information theory, pages 469–472, Volume 31, 1985.
2. J. (Benaloh) Cohen and M. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In: Proceedings of the 26th Symposium on Foundations of Computer Science, pages 372–382, 1985.
3. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings on advances in cryptology (CRYPTO '86), pages 186–194, 1986.
4. J. Benaloh and M. Yung. Distributing the Power of a Government to Enhance the Privacy of Voters In: Proceedings of the 5th Symposium on Principles of Distributed Computing, pages 52–62, 1986.
5. J. Benaloh. Verifiable Secret-Ballot Elections. Yale University Department of Computer Science Technical Report number 561, 1987.
6. M.Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. In: 20th STOC, pages 103–112, 1988.
7. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty-second annual ACM symposium on theory of computing (STOC '90), pages 42–437, 1990.
8. C. Schnorr. Efficient signature generation for smart cards. In: Journal of cryptology, Volume 4, pages 161–174, 1991.

9. I. Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In Eurocrypt, volume 658 of LNCS, pages 341–355, 1992.
10. M. Bellare and P. 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, 1993.
11. J. Benaloh and D. Tuinstra. Receipt-Free Secret-Ballot Elections. In: Proceedings of the 26th ACM Symposium on Theory of Computing, pages 544–553, 1994.
12. R. Gennaro. Achieving independence efficiently and securely. In: Proceedings of the 14th Principles of Distributed Computing Symposium (PODC'95), pages 130–136, 1995.
13. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In: Advances in Cryptology (EUROCRYPT '97), pages 256–266, 1997.
14. R. Cramer, R. Gennaro and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Advances in Cryptology (EUROCRYPT '97), pages 103–118, 1997.
15. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In: Proceedings of the 17th annual international cryptology conference on advances in cryptology (CRYPTO '97), pages 410–424, 1997.
16. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Agains Chosen-Ciphertext Attack. In: Advances in Cryptology (Eurocrypt '98), LNCS 1403, pages 1–16, 1998.
17. Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In: International Workshop on Practice and Theory in Public Key Cryptography (PKC '98), pages 117–134, 1998.
18. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proceedings of th 40th annual symposium on foundations of computer science (FOCS '99), pages 543–553, 1999.
19. C.P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '00), pages 73–89, 2000.
20. M. Bellare, A. Boldyreva and J. Staddon. Multi-recipient encryption schemes: Security notions and randomness re-use. Full version, `http://cseweb.ucsd.edu/~mihir/papers/bbs.html`. Preliminary version in: Public key cryptography (PKC 2003), Lecture notes in computer science, Vol. 2567, 2003.
21. J. Groth. Evaluating Security of Voting Schemes in the Universal Composability Framework. Applied Cryptography and Network Security, ACNS 2004, pages 46–60, 2004.
22. M. Fischlin. Communication-Efficient Non-Interactive Proofs of Knowledge with Online Extractors. In: Proceedings of the 25th annual international cryptology conference on advances in cryptology (CRYPTO '05), pages 152–168, 2005.
23. A. Juels, D. Catalano and M. Jakobsson. Coercion-Resistant Electronic Elections. In: Proceedings of the 4th Workshop on Privacy in the Electronic Society (WPES'05), pages 61–70, 2005.
24. S. Kremer and M. D. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In: 14th European Symposium on Programming (ESOP '05), pages 186–200, 2005.
25. T. Moran and M. Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: Proceedings of the 26th International Cryptology Conference (CRYPTO'06), pages 373–392, 2006.

26. S. Delaune, S. Kremer and M. D. Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. 19th Computer Security Foundations Workshop (CSFW '06), pages 28–42, 2006.
27. B. Chevallier-Mames, P. Fouque, D. Pointcheval, J. Stern and J. Traoré. On Some Incompatible Properties of Voting Schemes. In: Proceedings of the Workshop on Trustworthy Elections (WOTE'06), 2006.
28. Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl Accord. `http://www.dagstuhlaccord.org/`, 2007.
29. J. Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In: Proceedings of the Second Usenix/ACCURATE Electronic Voting Technology Workshop, 2007.
30. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '98), pages 13–25, 2008.
31. M. R. Clarkson, S. Chong and A. C. Myers, Civitas: Toward a Secure Voting System. In: Proceedings of the 29th Security and Privacy Symposium (S&P'08), pages 354–368, 2008.
32. B. Adida. Helios: Web-based open-audit voting. In: 17th USENIX security symposium, pages 335–348, 2008. `http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf`
33. M. Backes, C. Hriţcu and M. Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08), pages 195–209, 2008.
34. D. Wikström. Simplified Submission of Inputs to Protocols. In: Security and Cryptography for Networks, 6th International Conference, SCN 2008, pages 293–308, 2008.
35. B. Adida, O. de Marneffe, O. Pereira and J.-J. Quisquater. 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.
36. International association for cryptologic research. Election page at `http://www.iacr.org/elections/2010`
37. V. Cortier and B. Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. Website with description and video at `http://www.bensmyth.com/publications/10-attacking-helios/` Cryptology ePrint Archive, Report 2010/625.
38. S. Kremer, M. D. Ryan and B. Smyth. Election verifiability in electronic voting protocols. In: Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10), pages 389–404, 2010.
39. D. Unruh and J. Müller-Quade. Universally Composable Incoercibility. In: Proceedings of the 30th International Cryptology Conference (CRYPTO'10), pages 411–428, 2010.
40. R. Küsters, T. Truderung and A. Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. In: Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10), pages 122–136, 2010.
41. J. Loftus, A. May, N.P. Smart and F. Vercauteren. On CCA-Secure Fully Homomorphic Encryption `http://eprint.iacr.org/2010/560`
42. V. Cortier and B. Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. To appear in: Proceedings of the 24th Computer Security Foundations Symposium (CSF '11), 2011.

43. R. Küsters, T. Truderung and A. Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. Preprint, to appear at the 32nd Security and Privacy Symposium (S&P'11).
44. G. Persiano. About the Existence of Trapdoors in Cryptosystems. "Work in Progress", Available at http://libeccio.dia.unisa.it/Papers/Trapdoor/.
45. Helios voting. Website: http://heliosvoting.org
46. Helios Headquarters, Princeton University Undergraduate Student Government. http://usg.princeton.edu/officers/elections-center/helios-headquarters.html

A Details of our construction

We give a detailed description of our proposed construction of a provably secure extension of Helios.

A.1 Choice of groups

Let λ be a security parameter. Pick primes p, q such that $p - 1 = kq$ for some $k \in \mathbb{Z}$ and $q \approx \lambda$. Pick an element $g \in (\mathbb{Z}_p^*, \times)$ of order q. This defines a cyclic group of prime order q which we will call G. The group operation is integer multiplication with reduction mod p.

Occasionally we need the group $(\mathbb{Z}_q, +)$ directly. In this group the operation is integer addition with reduction mod q. Although these groups are isomorphic, for security we rely on the fact that we have a homomorphism $\mathbb{Z}_q \to G$ and the conjecture that it is hard to find one in the opposite direction.

The choice of these groups and parameters can be done as in existing Helios implementations. We summarize these groups in the following table.

Group	Order	Op.	Modulus	Neutral	Generator
G	q	$\times$	p	1	g
$\mathbb{Z}_q$	q	$+$	q	0	1

A.2 Key generation

An election is created by a number of trustees who each hold a share of the decryption/tallying key. To create an election for s trustees, each trustee $\mathcal{T}_i$ picks two secrets $x_{(i,0)}, x_{(i,1)} \xleftarrow{R} \mathbb{Z}_q$ where $i \leftarrow 1 \ldots s$. Each trustee then computes two values $y_{(i,b)} = g^{x_{(i,b)}} \mod p$ for $b \in \{0, 1\}$ and publishes these values. The election public key is the pair

$$\left(y_0 = \prod_{i=1}^{s} y_{(i,0)},\ y_1 = \prod_{i=1}^{s} y_{(i,1)} \right)$$

A trustee must also prove knowledge of his secrets with a proof of knowledge of a discrete logarithm. Although the proof of correct decryption of the election result will also include a proof of knowledge of the trustees' secret keys, voters must be able to verify that the election public keys are well-formed before they use them to encrypt their votes. The proof can be done as follows by trustee $\mathcal{T}_i$ and repeated for both $j = 0$ and $j = 1$. It is a straightforward Schnorr proof:

1. $\alpha_j \xleftarrow{R} \mathbb{Z}_q$.
2. $\beta_j \leftarrow g^{\alpha_j} \mod p$.
3. $\gamma_j \leftarrow H(\beta_j)$.
4. $\delta_j \leftarrow \alpha_j + \gamma_j \cdot x_{(i,j)} \mod q$.

The proof is the tuple $(\beta_0, \delta_0, \beta_1, \delta_1)$ and can be checked by ensuring $\beta_j \in G, \delta_j \in \mathbb{Z}_q$ and $g^{\delta_j} = \beta_j \cdot y_{(i,j)}^{\gamma_j} \mod p$ for $j \in \{0, 1\}$.

A.3 Creating a ballot

A ballot encodes a number of votes. The election parameters describe

- The number n_V of votes each voter must cast.
- For each vote i in $1 \dots n_V$, a range $(\min_i, \max_i)$ that the vote must lie in.
- A range $(\min_V, \max_V)$ that the sum of all votes must lie in.

To create a ballot for votes $v_1 \dots v_{n_V}$ satisfying these conditions, a voter performs the following algorithm. In the interest of readability, we split the algorithm into several distinct procedures as is good practice in programming and give comments and pre/postconditions for each.

create-ballot This is the main algorithm that takes a list of votes and creates a ballot.
Input: Number of votes n_V, votes $v_1 \dots v_{n_V}$, ranges $(\min_i, \max_i)_{i=1}^{n_V}$, range $(\min_V, \max_V)$, public key (y_0, y_1).
Preconditions: $n_V \geq 1$, for all $i \in 1 \dots n_V$ we have $\min_i \leq \max_i$, $\min_V \leq \sum_{i=1}^{n_V} \min_i \leq \sum_{i=1}^{n_V} \max_i \leq \max_V$.
Output: Valid ballot for $v_1 \dots v_{n_V}$.

1. For $i \leftarrow 1 \dots n_V$:
   (a) $r_i \xleftarrow{R} \mathbb{Z}_q$.
   (b) $e_i \leftarrow \mathsf{encrypt\text{-}vote}(y_0, y_1, v_i, r_i)$.
   (c) $\pi_{i,NY} \leftarrow \mathsf{create\text{-}NY\text{-}proof}(y_0, y_1, r_i, e_i)$.
   (d) $\pi_{i,R} \leftarrow \mathsf{create\text{-}range\text{-}proof}(y_0, y_1, r_i, v_i, e_i, \min_i, \max_i)$.
2. $\pi_V \leftarrow \mathsf{create\text{-}sum\text{-}proof}(y_0, y_1, (e_i, r_i, v_i)_{i=1}^{n_V}, \min_V, \max_V)$.
3. Return the ballot
$$((e_i, \pi_{i,NY}, \pi_{i,R})_{i=1}^{n_V}, \pi_V)$$

encrypt-vote This procedure creates a 3-element ElGamal encryption of a vote.
Input: Vote v, randomness r, public keys y_0, y_1.
Preconditions: None.
Output: Encrypted vote $e = (a, b, c)$.
Compute
$$(a \leftarrow g^r,\ b \leftarrow g^v y_0^r,\ c \leftarrow g^v y_1^r) \mod p$$
and return $e \leftarrow (a, b, c)$.

create-NY-proof This procedure takes an encrypted vote e and the underlying randomness r and produces a proof that both encryptions are for the same message under the same randomness.
Input: Public keys y_0, y_1, randomness r, encrypted vote e.
Preconditions: e is a valid encryption of a vote with randomness r under public keys y_0, y_1.
Output: Non-interactive zero-knowledge proof π.

1. $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_q$.
2. $(\beta_1 \leftarrow g^{\alpha_2},\ \beta_2 \leftarrow g^{\alpha_1} y_0^{\alpha_2},\ \beta_3 \leftarrow g^{\alpha_1} y_1^{\alpha_2}) \mod p$.
3. $\gamma \leftarrow H(\beta_1 \| \beta_2 \| \beta_3)$.
4. $\delta_1 \leftarrow \alpha_1 + \gamma \cdot m \mod q,\ \delta_2 \leftarrow \alpha_2 + \gamma \cdot r \mod q$.
5. Return $\pi \leftarrow (\beta_1, \beta_2, \beta_3, \delta_1, \delta_2)$.

create-range-proof This procedure produces a zero-knowledge proof that a vote lies in a given range. Note that we only consider the first public key which is sufficient as the zero-knowledge proof of equality guarantees that the second encryption will be in the same range.
Input: Public keys y_0, y_1, encrypted vote e, vote v, randomness r, range of valid votes $(\min, \max)$.
Preconditions: e is the encryption of v under public keys y_0, y_1 and randomness r and $v \in [\min, \max]$.
Output: Zero-knowledge proof π.

1. Parse e as (a, b, c).
2. For $j \leftarrow [\min, \max] \setminus \{v\}$, create a simulated proof:
   (a) $\gamma_j, \delta_j \xleftarrow{R} \mathbb{Z}_q$.
   (b) $\Delta_{j,1} \leftarrow g^{\delta_j} \mod p,\ \Delta_{j,2} \leftarrow y_0^{\delta_j} \mod p$.
   (c) $\beta_{j,1} \leftarrow \Delta_{j,1} - \gamma_i \cdot a \mod p,\ \beta_{j,2} \leftarrow \Delta_{j,2} - \gamma_i \cdot (b/g^j) \mod p$.
3. $\alpha_v \xleftarrow{R} \mathbb{Z}_q$.
4. $\beta_{v,0} \leftarrow g^{\alpha_v} \mod p,\ \beta_{v,1} \leftarrow y_0^{\alpha_v} \mod p$.
5. $\gamma \leftarrow H(\beta_{\min,1} \| \beta_{\min,2} \| \beta_{\min+1,1} \| \beta_{\min+1,2} \| \ldots \| \beta_{\max,1} \| \beta_{\max,2})$.
6. $\gamma_v \leftarrow \gamma - \sum_{j=\min, j \neq v}^{\max} \gamma_j \mod q$.
7. $\delta_v \leftarrow \alpha_v + \gamma_v \cdot r$.
8. Return $(\beta_{j,1}, \beta_{j,2}, \gamma_j, \delta_j)_{j=\min}^{\max}$.

create-sum-proof This procedure produces a zero-knowledge proof that the sum of all votes $(v_1, \ldots, v_{n_V})$ underlying a single ballot lies in the range $[\min_V, \max_V]$. This is done by homomorphically adding all votes and then producing a range proof on the sum. Sum proofs, like range proofs, need only be done on one of the two encryption.
Input: Public keys y_0, y_1, encrypted votes, underlying randomness and plain votes $(e_i, r_i, v_i)_{i=1}^{n_V}$, range $(\min_V, \max_V)$.
Preconditions: Valid encrypted votes for given public keys, plain votes and randomness; sum of all plain votes lies in range $[\min_V, \max_V]$.
Output: Zero-knowledge proof π.

1. Parse each encrypted vote as (a_i, b_i, c_i).
2. $(a, b) \leftarrow (\prod_{i=1}^{n_V} a_i \mod p,\ \prod_{i=1}^{n_V} b_i \mod p)$.
3. $(r, v) \leftarrow (\sum_{i=1}^{n_V} r_i \mod q, \sum_{i=1}^{n_V} v_i \mod q)$.
4. Return $\mathsf{create\text{-}range\text{-}proof}(y_0, y_1, r, v, (a, b, \perp), \min_V, \max_V)$.

A.4 Verifying a ballot

When a ballot is cast, the entity responsible for collecting ballots must check its validity and publish it if found valid. This process must be publicly verifiable, indeed anyone must be able to check ballot validity without access to any secret information.

Ballot authentication is out of scope of this discussion, for simplicity we may assume authentic channels from all voters to the ballot casting centre. We explicitly include checks that all elements are in the required groups.

verify-ballot This algorithm takes a ballot and returns 1 if it is deemed to be valid, otherwise 0.
Input: A claimed ballot of the form $((e_i, \pi_{i,NY}, \pi_{i,R})_{i=1}^{n_V}, \pi_V)$.
Preconditions: None (we are checking if the input is valid, after all).
Output: 0 or 1.

Wherever "check" is written, the algorithm immediately aborts returning 0 if a check fails.

1. Check that the number of elements is correct.
2. For $i \leftarrow 1 \dots n_V$
   (a) Parse e_i as (a, b, c). Check all three elements are in G.
   (b) Check verify-NY$(y_0, y_1, e_i, \pi_{i,NY})$.
   (c) Check verify-range$(y_0, y_1, e_i, \pi_{i,R})$.
3. $(a', b') \leftarrow (\prod_{i=1}^{n_V} a_i, \prod_{i=1}^{n_V} b_i) \mod p$.
4. Check verify-range$(y_0, y_1, (a', b', \perp), \pi_V)$.

verify-NY This procedure verifies a Naor-Yung transformation proof that two encryptions share the same message and randomness.
Input: Public key (y_0, y_1), encrypted vote $e = (a, b, c)$, Naor-Yung proof $\pi = (\beta_1, \beta_2, \beta_3, \delta_1, \delta_2)$.
Preconditions: $a, b, c \in G$.
Output: 0 or 1.

1. Check that $\beta_1 \in G, \beta_2 \in G, \beta_3 \in G, \delta_1 \in \mathbb{Z}_q, \delta_2 \in \mathbb{Z}_q$ or return 0 if this fails.
2. $\Delta_1 \leftarrow g^{\delta_2} \mod p$, $\Delta_2 \leftarrow g^{\delta_1} y_0^{\delta_2} \mod p$, $\Delta_3 \leftarrow g^{\delta_1} y_1^{\delta_2} \mod p$.
3. $\gamma \leftarrow H(\beta_1 \| \beta_2 \| \beta_3)$.
4. Return 1 if the following checks pass, otherwise 0:
   - $\Delta_1 \stackrel{?}{=} \beta_1 \cdot a^{\gamma} \mod p$.
   - $\Delta_2 \stackrel{?}{=} \beta_2 \cdot b^{\gamma} \mod p$.
   - $\Delta_3 \stackrel{?}{=} \beta_3 \cdot c^{\gamma} \mod p$.

verify-range This procedure verifies a zero-knowledge proof that an encrypted vote lies within a given range.

Input: Public key (y_0, y_1), encrypted vote (a, b, c), disjunctive proof $(\beta_{j,1}, \beta_{j,2}, \gamma_j, \delta_j)_{j=\min}^{\max}$.
Preconditions: $a, b, c \in G$, encrypted vote NY-verified.
Output: 0 or 1.

1. For $j \leftarrow \min, \ldots, \max$:
   (a) Check that $\beta_{j,1} \in G, \beta_{j,2} \in G, \gamma_j \in \mathbb{Z}_q, \delta_j \in \mathbb{Z}_q$ or return 0 if this fails.
   (b) $\Delta_{j,1} \leftarrow g^{\delta_j} \mod p$, $\Delta_{j,2} \leftarrow y_0^{\delta_j} \mod p$.
   (c) Return 0 if any of the following two checks fail:
      – $\Delta_{j,1} \stackrel{?}{=} \beta_{j,1} \cdot a^{\gamma_j} \mod p$.
      – $\Delta_{j,2} \stackrel{?}{=} \beta_{j,2} \cdot (b/g^j)^{\gamma_j} \mod p$.
2. $\gamma \leftarrow H(\beta_{\min,1} \| \beta_{\min,2} \| \beta_{\min+1,1} \| \beta_{\min+1,2} \| \ldots \| \beta_{\max,1} \| \beta_{\max,2})$.
3. Check that $\gamma \stackrel{?}{=} \sum_{j=\min}^{\max} \gamma_j$. Return 1 if this succeeds, otherwise 0.

A.5 Computing a Tally

A group of trustees each hold a share of the decryption key. Once voting is over, they must each verify all ballots and reject any that fail the checks. It is also advisable that they check for duplicate proofs among all valid ballots. After this stage, all zero-knowledge proofs and even the third elements of triples forming encrypted votes can be discarded.

Each trustee should independently compute the sum of all votes. Specifically, if the ballot of voter i contains the n_V encryptions $(a_{i,j}, b_{i,j})_{j=1}^{n_V}$ then the trustees should compute

$$\left(a_j \leftarrow \prod_{i=1}^{n} a_{i,j},\ b_j \leftarrow \prod_{i=1}^{n} b_{i,j} \right)_{j=1}^{n_V}$$

Although this must be publicly computable to verify the election, it is important that each trustee does not apply his decryption key to anything until he is sure it is the correct sum.

The trustees can each compute a partial decryption by applying their secret key share: The trustee holding x_k (which we can take to be $x_{(k,0)}$) computes $(d_{j,k} \leftarrow (a_j)^{x_k} \mod p)_{j=1}^{n_V}$ and publishes this value along with a proof of knowledge attesting to the fact that they decrypted correctly. Such a proof (to be repeated for $j \leftarrow 1 \ldots n_V$) takes the form

$$\mathbf{PoK}\{(x_k) : d_{j,k} = a_j^{x_k} \mod p \ \wedge\ y_k = g^{x_k} \mod p\}$$

and is computed as follows.

1. $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_q$
2. $(\beta_1, \beta_2) \leftarrow (a_j^{\alpha}, g^{\alpha}) \mod p$.
3. $\gamma \leftarrow H(\beta_1 \| \beta_2)$.
4. $\delta \leftarrow \alpha + \gamma \cdot x_k \mod q$.

The proof consists of the elements $(\beta_1, \beta_2, \delta)$ and verification involves checking the following three conditions.

1. β_1 and β_2 are in G and δ in $\mathbb{Z}_q$.
2. $(a_j)^\delta = \beta_1 \cdot d_{j,k}^{H(\beta_1 \| \beta_2)} \mod p$.
3. $g^\delta = \beta_2 \cdot y_k^{H(\beta_1 \| \beta_2)} \mod p$.

The decrypted tally is then $(d_j \leftarrow b_j / \prod_{k=1}^{n_K} d_{j,k})_{j=1}^{n_V}$ where n_K is the number of key shares. This will satisfy the equation $(d_j = g^{m_j} \mod p)$ where m_j is the sum of all jth votes (for $j \leftarrow 1 \ldots n_V$). The actual result m_j can be recovered by computing powers of g modulo p until all values d_j are found.

A.6 Simulating a Tally

In the ballot privacy game for $b = 1$, we need to simulate these proofs. The values $d_{j,k}$ must be computed correctly and the proof simulated as follows.

1. $\gamma \xleftarrow{R} \mathbb{Z}_q,\ \delta \xleftarrow{R} \mathbb{Z}_q$
2. $\beta_1 \leftarrow (a_j)^\delta / (d_{j,k})^\gamma \mod p$
3. $\beta_2 \leftarrow (g)^\delta / (y_k)^\gamma \mod p$
4. Patch H at input $(\beta_1 \| \beta_2)$ to equal γ.

A.7 Verifying the Election

Anyone can publicly perform the following verification steps.

- Check each published ballot for validity.
- Check there are no duplicate proofs among published ballots.
- Compute the encryption of the tally from the ballots using the homomorphic property of the ElGamal encryption.
- Check the proofs of correct partial decryption.
- Compute the tally from the partial decryptions.

B Ballot Privacy for the Mini–Voting Scheme

For the proof, we first consider a "mini-voting" scheme (in Theorem 4) that removes all the extra zero-knowledge proofs concerning verifiability and show that ballot privacy follows from IND-CCA2 security of the encryption scheme. Then, we extend this proof to our construction (in Theorem 1).

We stress that we do not require any additional conditions on the zero-knowledge proofs (for verifiability); in particular, they need not be non-malleable.

Theorem 4. *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CCA2 secure encryption scheme for message space* $\mathbb{V}$*. Then the mini-voting scheme in Algorithm 7 is a voting scheme with ballot privacy.*

Algorithm 7 Mini-Voting Scheme

Setup
  $(x, y) \leftarrow \mathsf{Gen}$
  $\mathcal{Z} \leftarrow y$
  $BB \leftarrow ()$
Vote(v)
  $b \leftarrow \mathsf{Enc}(v)$
ProcessBallot(c, BB)
  if $b \notin BB$ **then**
    $BB \xleftarrow{+} b$
    return ("accept", BB)
  else
    return ("reject", BB)
  end if
Tally(x, BB)
  for all $b_j \in BB$ **do** $v'_j \leftarrow \mathsf{Dec}(b_j)$ **end for**
  $r \leftarrow \rho((v'_j)_{j=1}^{|BB|})$

Proof (Theorem 4). Suppose $\mathcal{A}$ is an adversary that has non-negligible advantage in winning the ballot privacy game.

Let n be an upper bound on the number of vote calls the adversary makes. Let $(H_k)_{k=0}^{n}$ be the sequence of games where in game H_k, the first k calls to vote are answered as if $\beta = 0$ and all from the $k+1$st onwards are answered as if $\beta = 1$. We note that H_0 is equivalent to the ballot privacy game for $\beta = 0$ and H_n for $\beta = 1$.

Using the triangular inequality, if any adversary can distinguish H_0 from H_n with non-negligible advantage then there is a value k such that the adversary can distinguish the pair of hybrids (H_k, H_{k+1}) with non-negligible advantage.

Given such a k, we construct a reduction to IND-CCA2 of the underlying scheme. We assume w.l.o.g. that in the $k+1$st call to vote, the vote is not ε or else the two hybrids would be identical. Let $\mathcal{C}$ be a challenger for IND-CCA2 of the encryption scheme. We construct an adpater $\mathcal{B}$ which will play against $\mathcal{C}$ using $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ receives a public key y from $\mathcal{C}$, performs the setup of the voting scheme and hands the public key and parameters to $\mathcal{A}$.

vote

queries $1 \ldots k$**:** $\mathcal{B}$ processes these as in the ballot privacy game with $\beta = 0$.

query $k+1$**:** Let v^* be the input to this query. If it is $\perp$, $\mathcal{B}$ does nothing. If not, it stores v^* and forwards the pair (v^*, ε) to $\mathcal{C}$ and gets a ciphertext c^* back. It processes c^* onto BB and creates a new ciphertext $c' = \mathsf{Vote}(v^*)$ which it processes onto BB'. Finally, $\mathcal{B}$ returns the new state of BB to $\mathcal{A}$.

queries $\geq k+2$**:** $\mathcal{B}$ processes these as in the ballot privacy game with $\beta = 1$.

ballot
: For all queries before the $k+1$st vote query (when $\mathcal{B}$ obtains the IND-CCA2 challenge), it passes the ballot to the decryption oracle and stores the plaintext and ballot. $\mathcal{B}$ then handles the query like the ballot privacy experiment.

Tallying As long as c^* does not appear on BB', $\mathcal{B}$ decrypts all ballots on BB' (using the decryption oracle of $\mathcal{C}$) to get the votes, evaluates ρ on these votes and returns the result to $\mathcal{A}$. We deal with the case that c^* is on BB' below.

Guess $\mathcal{B}$ forwards the bit $\widehat{\beta}$ it receives from $\mathcal{A}$ to $\mathcal{C}$.

As long as c^* does not appear on BB', the adversary's view when interacting with $\mathcal{B}$ will be identically distributed to that when he is interacting with $H_{k+\beta}$ where β is the bit chosen by $\mathcal{C}$ and therefore $\mathcal{B}$ wins against $\mathcal{C}$ with the same advantage as $\mathcal{A}$ has in distinguishing the two hybrids.

Suppose c^* appears on BB' and consider the following cases. In each case, we show that $\mathcal{B}$ obtains the plaintext of c^* and can therefore guess β with probability 1 as we assumed $v^* \neq \varepsilon$.

1. c^* was added while processing a ballot command *before* the challenge query. In this case, $\mathcal{B}$ has decrypted c^* and stored the plaintext/ciphertext pair while processing the command.
2. c^* was added while processing a ballot command *after* the challenge query. This is impossible as c^* is definitely on BB after the challenge query and so further calls to ballot(c^*) would reject before trying to process c^* onto BB'.
3. c^* was added while processing a vote query.
   In this case, $\mathcal{B}$ knows the plaintext because it receives it as input.

Therefore, $\mathcal{B}$ has at least as high an advantage against $\mathcal{C}$ as $\mathcal{A}$ has of distinguishing H_k from H_{k+1}. □

Proof (Theorem 1). We will prove the theorem by a sequence of "game hops". Starting with the mini-voting scheme, we modify the voting scheme in each hop and argue how the proof needs to be adapted.

Let *Scheme 2* be the mini-voting scheme where each voter may cast k votes. Duplicate checking is performed as in our full voting scheme, i.e. we extract the list of ciphertexts from each ballot and reject if any of them is a duplicate. For n voters, this gives us a total of $N := n \cdot k$ encrypted votes and we simply run the same hybrid argument (with N hybrids).

Next, we add the proofs of knowledge everywhere to obtain *Scheme 3*. In the security reduction, we perform two hybrid arguments. The first one replaces the zero-knowledge proofs by simulated ones, one step at a time. The second one switches votes to ε as before. If the adversary can detect a difference between any two consecutive hybrids, he can either attack the encryption (as argued earlier) or distinguish a real from a simulated zero-knowledge proof.

Finally, we switch the tallying operation to perform checking, extraction, homomorphic tallying and decryption of the result only to obtain our proposed scheme. (If ρ is not homomorphically computable, we skip this step as our scheme

cannot tally homomorphically either.) We extend our sequence of games by 1, where the first hop is switching from homomorphic decryption back to decrypting all ballots. The adversary will not notice any difference between the two games as he is not involved in the decryption and tallying process, so ballot privacy still holds.

□

C Security proof of the Naor-Yung transformation

C.1 The Naor-Yung Transformation

We prove security of the Naor-Yung transformation [7].

Naor and Yung originally used zero-knowledge proofs of language membership yet their technique and proof can easily be adapted to proofs of knowledge. This adaptation also removes a substantial amount of the complexities they had to deal with in their original paper.

Naor and Yung proved IND-CCA1 of their transformation. Sahai [18] realized that this transformation even gives an IND-CCA2 secure encryption scheme if the zero-knowledge proof system satisfies a few extra conditions. All of these hold trivially for proofs of knowledge except the one he called *uniquely applicable proofs*; we will have to show that this holds for our proposed scheme.

We prove IND-CCA2 security of the Naor-Yung transformation two steps. First, we prove Naor and Yung's theorem [7] giving us IND-CCA1 security. The original theorem concerned zero-knowledge proofs of *language membership* and used specific definitions and propositions concerned with such proofs. Our version omits all these as we deal with the simpler case of proofs of *knowledge*. The main ideas in the proof remain unchanged although we use techniques from [20] to simplify the presentation.

Proof (Naor-Yung). Let $\mathcal{C}$ be a challenger for the IND-CPA game of the original encryption and $\mathcal{A}$ be an adversary for the IND-CCA1 game of the Naor-Yung transformed scheme. We construct an algorithm $\mathcal{B}$ that attacks $\mathcal{C}$ using $\mathcal{A}$.

When $\mathcal{B}$ is invoked by $\mathcal{A}$, it acts as follows.

Setup
1. Receive a public key y from $\mathcal{C}$.
2. Create a key pair (x', y') using Gen.
3. Pick a bit $b' \xleftarrow{R} \{0,1\}$.
4. If $b' = 0$, let $(y_1, y_2) \leftarrow (y, y')$. Otherwise, let $(y_1, y_2) \leftarrow (y', y)$ i.e. b' determines the order of the keys.
5. Send (y_1, y_2) to $\mathcal{A}$.

$\mathsf{Dec}(\mathsf{c})$ Parse c as (c_0, c_1, π). Check π is valid for c_0, c_1 and abort returning $\perp$ if this fails. Decrypt $c_{1-b'}$ with x' and return the result m.

$\mathsf{Chal}(\mathsf{m}_0, \mathsf{m}_1)$
1. Pass (m_0, m_1) to the Chal interface of $\mathcal{C}$ and get a ciphertext c back.
2. Create a new ciphertext $c' \leftarrow \mathsf{Enc}(y', m_{b'})$.

3. If $b' = 0$, let $(c_0, c_1) \leftarrow (c, c')$. If $b' = 1$, swap the order of ciphertexts: $(c_0, c_1) \leftarrow (c', c)$.
4. Simulate a proof $\pi \leftarrow \mathsf{Sim}(c_0, c_1)$.
5. Return (c_0, c_1, π) to the adversary.

Result When $\mathcal{A}$ produces a result $d \in \{0, 1\}$ forward this to $\mathcal{C}$.

The public keys in **Setup** are distributed identically to those in the Naor-Yung transformed encryption scheme. The decryption oracle will work correctly for $\mathcal{A}$ as long as he does not manage to forge a proof and submit an invalid ciphertext; the probability of this happening is negligible as we assume the proof system to be sound.

Consider the distributions $\mathcal{A}$ sees based on the bits b of $\mathcal{C}$ and b' of $\mathcal{B}$. For $x, y \in \{0, 1\}$ define the distribution

$$\delta(x, y) := (\mathsf{Enc}(y_0, m_x), \mathsf{Enc}(y_1, m_y))$$

over the random choices in both encryptions, assuming the public keys and messages are fixed. $\mathcal{A}$ sees two elements (c_0, c_1) and a proof π in response to his **Chal** query. The distribution of these ciphertexts will be

	$b = 0$	$b = 1$
$b' = 0$	$\delta(0, 0)$	$\delta(1, 0)$
$b' = 1$	$\delta(1, 0)$	$\delta(1, 1)$

The advantage of $\mathcal{A}$ against IND-CCA1 security of the Naor-Yung transformed scheme is his advantage in distinguishing $\delta(0, 0)$ from $\delta(1, 1)$. If this is non-negligible, then by the triangular inequality his advantage between two successive distributions in the sequence

$$\delta(0, 0) \;-\; \delta(1, 0) \;-\; \delta(1, 1)$$

will also be non-negligible. In fact, as $\mathcal{B}$ chooses b' uniformly at random from $\{0, 1\}$ we can estimate

$$\mathbf{Adv}^{\mathsf{CPA}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathsf{CCA1}}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{P}}(\mathcal{A})$$

Where $\mathbf{Adv}^{\mathsf{P}}$ is the probability that $\mathcal{A}$ can attack the zero-knowledge proof, either by forging a proof for an incorrect encryption or by distinguishing a real proof from a simulated one (on a correct or incorrect encryption). Assuming the proof system is sound and zero-knowledge, this quantity is negligible. □

We now prove Sahai's version of the theorem that gives us IND-CCA2 security. Sahai proved the theorem directly whereas we just need to extend from IND-CCA1 to IND-CCA2 security. Sahai used proofs of language membership, as we consider proofs of knowledge we can again omit many of the technicalities. The central argument remains the same, namely showing that the adversary cannot ask for a decryption of an invalid ciphertext.

Proof (Sahai). We claim that for any triple (c_0, c_1, π) submitted to the decryption oracle, if the proof verifies then with overwhelming probability both c_0 and c_1 are encryptions of the same message.

There is only one case when the adversary sees a proof he has not created himself (so we cannot use the extractor of the proof of knowledge to get a contradiction) and that is the proof returned from the challenge query.

Uniquely applicable proofs give us that the adversary cannot create a triple (c_0^*, c_1^*, π) distinct from the response of the challenge oracle, passing verification using the same proof π. In the definition of IND-CCA2, the adversary is not allowed to ask for a decryption of the triple he got from the challenge oracle either.

For any triple with a proof $\pi^* \neq \pi$ distinct from that we returned from the challenge query, if the underlying statement is false then the adversary has produced a forgery which we assume can happen with at most negligible probability. □

(The property that the adversary may not prove a false statement even after seeing a proof for a false statement is called simulation-soundness in the literature. For a proof of knowledge with uniquely applicable proofs, this is trivial.)

We can now prove our theorem that we get a voting-friendly scheme.

Proof (Theorem 3). IND-CCA2 follows from Sahai's theorem. The ExtractKey and Extract algorithms extract the first key pair and ciphertext respectively; this yields the embedding property. EAdd is just the homomorphic addition algorithm of the original scheme which clearly can still be applied.

D Reusing randomness

Recall that a homomorphic ElGamal ciphertext is a pair of the form $(g^r, g^m y^r)$ mod p. For the Naor-Yung construction, we could simply use a 4-tuple of the form

$$(g^{r_1}, g^m y_1^{r_1}, g^{r_2}, g^m y_2^{r_2}) \mod p$$

but in fact we can do better thanks to a theorem of Bellare, Boldyreva and Staddon [20] which allows us to reuse the random value r.

We state the crucial condition for randomness reuse and the theorem informally and without proof, as it covers a much more general case than we require.

An encryption scheme is said to be *reproducible* if there is an efficient algorithm Reproduce that, on input a public key and an encryption of some unknown message under this public key together with a new message and a new public/secret key pair, produces an encryption of the new message under the new public key using *the same randomness* as the old encryption.

More formally, let $\mathsf{Enc}(pk, m; r)$ denote the *deterministic* encryption function, where r is the randomness. Then we demand $\forall (pk, sk), m, r, (pk', sk')$ where (pk, sk) and (pk', sk') are valid key pairs

$$c = \mathsf{Enc}(pk, m; r) \rightarrow \mathsf{Reproduce}(pk, c, m', pk', sk') = \mathsf{Enc}(pk', m'; r)$$

ElGamal is reproducible: Given a ciphertext $c = (a, b)$ we can reproduce one for a new key as

$$\mathsf{Reproduce}(pk, c, m', pk', sk') := (a, g^{m'} a^{sk'})$$

Writing $a = g^r, b = g^m pk^r$ we find that the new element is $g^{m'}(g^r)^{sk'} = g^{m'}(pk')^r$ as required.

Informally, the relevant theorem in [20] states that if a public-key encryption is reproducible and IND-CPA, CCA1 or CCA2 then it retains this security if encryption is performed with the same randomness for multiple keys. Their definition of multi-recipient IND-CPA/CCA also takes into account that the adversary may be a legitimate participant in the system and know some of the decryption keys too. We refer to the original paper for the precise definitions and the full proof, we do not require these for our construction.

An encryption of message m with randomness r under keys y_1, y_2 will be the triple

$$(g^r, g^m y_1^r, g^m y_2^r) \mod p$$

Apart from reducing bandwidth and computation required for the actual encryption compared to two separate encryptions, this also allows us to construct an easier zero-knowledge proof for the Naor-Yung transformation.

The proof of the Naor-Yung transformation is almost identical when reusing randomness, the only change is that we use $\mathsf{Randomize}$ in the challenge oracle to create the second ciphertext.

E Encryption

E.1 Public-Key Encryption

A public-key encryption scheme is a triple of probabilistic algorithms

$$(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$$

satisfying the correctness condition for all messages m in the message space with probability 1.

The algorithms and their input/output values are as follows.

$\mathsf{Gen}(1^\lambda)$ The key-generation algorithm takes a security parameter λ and generates a secret key x and a public key y.

$\mathsf{Enc}(y, m)$ The encryption algorithm takes a public key y and a message m and outputs a ciphertext c.

$\mathsf{Dec}(x, c)$ The decryption algorithm takes a secret key x and a ciphertext c and outputs a decryption d.

The scheme is IND-CPA, IND-CCA1 or IND-CCA2 secure if no efficient adversary can win the security game with non-negligible advantage, defined as

$$\mathbf{Adv}(\mathcal{A}) = \left| \mathbf{Pr}[\mathcal{A} \text{ wins}] - \frac{1}{2} \right|$$

Algorithm 8 Correctness of Public-Key Encryption

Parameters: m

$(x, y) \leftarrow \mathsf{Gen}$
$c \leftarrow \mathsf{Enc}(y, m)$
$m' \leftarrow \mathsf{Dec}(x, c)$
return $m = m'$

The idea underlying all three is that the adversary may pick any two plaintexts and submit them to a challenge oracle, receiving an encryption of one of the two in return. He must then decide which of the two messages the encryption represents. The difference between the three notions lies in when the adversary may make decryption queries:

Game	Before challenge call	After challenge call
IND-CPA	no	no
IND-CCA1	yes	no
IND-CCA2	yes	yes (*)

(*) The adversary may not ask for a decryption of the ciphertext he got from the challenge call.

Algorithm 9 Security of Public-Key Encryption

Adversary: $\mathcal{A}$

$(x, y) \leftarrow \mathsf{Gen}$
$\mathcal{A} \leftarrow y$
if IND-CCA1 or IND-CCA2 **then**
  while $c \leftarrow \mathcal{A}$ **do** $\mathcal{A} \leftarrow \mathsf{Dec}(c)$ **end while**
end if
$(m_0, m_1) \leftarrow \mathcal{A}$
$b \xleftarrow{R} \{0, 1\}$
$c^* \leftarrow \mathsf{Enc}(y, m_b)$
$\mathcal{A} \leftarrow c^*$
if IND-CCA2 **then**
  while $c \leftarrow \mathcal{A}$ **do**
    if $c = c^*$ **then** $\mathcal{A} \leftarrow \perp$ **else** $\mathcal{A} \leftarrow \mathsf{Dec}(c)$ **end if**
  end while
end if
$b' \leftarrow \mathcal{A}$
return $b = b'$

Security in any of these three models implies a probabilistic encryption function (and sufficient entropy), otherwise the adversary could just recompute encryptions of the two messages he sent to the challenge oracle himself and compare these with the ciphertext he received from it.

E.2 Homomorphic Encryption

A homomorphic public-key encryption scheme consists of four algorithms

$$(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$$

such that the message space is a group (denote the group operation by $\oplus$), the first three algorithms are a public-key encryption scheme and the fourth takes two ciphertexts and produces a ciphertext for the message corresponding to the group operation evaluated on the original two plaintexts, without access to the secret key. More formally, for any messages m_0, m_1 the additional correctness property is satisfied with probability 1.

Algorithm 10 Correctness of Homomorphic Encryption

Parameters: m_0, m_1

$(x, y) \leftarrow \mathsf{Gen}$
$c_0 \leftarrow \mathsf{Enc}(y, m_0)$
$c_1 \leftarrow \mathsf{Enc}(y, m_1)$
$c \leftarrow \mathsf{Add}(c_0, c_1)$
$m' \leftarrow \mathsf{Dec}(x, c)$
return $m' = m_0 \oplus m_1$

A homomorphic encryption scheme can never be IND-CCA2 secure. If the adversary chooses m_0, m_1 as his two challenge messages and gets a ciphertext c as a result, in a homomorphic scheme he can always pick any m' that is not the unit of the group, create $c' \leftarrow \mathsf{Enc}(y, m')$ and $c'' \leftarrow \mathsf{Add}(c, c')$. He can then send c'' to the decryption oracle (as the underlying plaintext differs from that of c, so must the ciphertext) to get m'' back and check if $m'' = m_0 + m'$ or $m'' = m_1 + m'$ holds.

E.3 Threshold Decryption

A public-key encryption scheme with *threshold decryption* for t out of n shares consists of algorithms

$$(\mathsf{Setup}, \mathsf{GenShare}, \mathsf{CombineKey}, \mathsf{Enc}, \mathsf{DecShare}, \mathsf{Combine})$$

for a given threshold t out of a number n of shares.

Setup The setup algorithm takes a security parameter as input and creates public parameters *params* which are implicitly available to all further algorithms.

GenShare This generates a pair (x, y) consisting of a secret key share and a public key share.

CombineKey This algorithm takes a list of n public key shares $(y_i)_{i=1}^n$ as input and returns a public key y.

Enc The encryption algorithm is identical to normal public-key encryption.

DecShare The shared decryption algorithm takes a private key share x_i and a ciphertext c as input and produces a partial decryption d_i.

Combine This takes a ciphertext c and a list of t partial decryptions and produces a plaintext.

The correctness game must return 1 with probability 1 for any message m and any set T of exactly t indices to use for decryption.

Algorithm 11 Correctness of Threshold Public-Key Encryption

Parameters: m, $T \subseteq \{1, \ldots, n\}$ with $|T| = t$

$params \leftarrow \mathsf{Setup}$
for $i \leftarrow 1 \ldots n$ **do** $(x_i, y_i) \leftarrow \mathsf{GenShare}$ **end for**
$y \leftarrow \mathsf{combineKey}((y_i)_{i=1}^n)$
$c \leftarrow \mathsf{Enc}(y, m)$
for all $i \in T$ **do** $d_i \leftarrow \mathsf{DecShare}(x_i, c)$ **end for**
$m' \leftarrow \mathsf{Combine}(c, (d_i, i)_{i \in T})$
return $m = m'$

For security, we allow the adversary to obtain any $t-1$ decryption key shares and give him partial decryption oracles for the others. The security notions are IND-TCPA, IND-TCCA1 and IND-TCCA2 where we demand that no efficient adversary can gain a non-negligible advantage in the game.

Definition 10 (Threshold Homomorphic Embedding). *A threshold scheme*

$$(\mathsf{Setup}, \mathsf{GenShare}, \mathsf{CombineKey}, \mathsf{Enc}, \mathsf{DecShare}, \mathsf{Combine})$$

has threshold homomorphic embedding

$$(\mathsf{ESetup}, \mathsf{EGenShare}, \mathsf{ECombineKey}, \mathsf{EEnc}, \mathsf{EDecShare}, \mathsf{ECombine}, \mathsf{EAdd})$$

if there are algorithms

$$(\mathsf{ExtractShare}, \mathsf{ExtractKey}, \mathsf{Extract}, \mathsf{EDecShare}, \mathsf{ECombine}, \mathsf{EAdd})$$

such that

$$\mathsf{EGenShare} = \mathsf{ExtractShare} \circ \mathsf{GenShare}$$

$$\mathsf{ECombineKey} = \mathsf{ExtractKey} \circ \mathsf{CombineKey}$$

$$\mathsf{EEnc} = \mathsf{Extract} \circ \mathsf{Enc}$$

$$\mathsf{Dec} = \mathsf{EDec} \circ \mathsf{Extract}$$

Algorithm 12 Security of Threshold Public-Key Encryption

Adversary: $\mathcal{A}$

$params \leftarrow \mathsf{Setup}$
$\mathcal{A} \leftarrow params$
for $i \leftarrow 1 \ldots n$ **do** $(x_i, y_i) \leftarrow \mathsf{GenShare}$ **end for**
$y \leftarrow \mathsf{CombineKey}((y_i)_{i=1}^n)$
$\mathcal{A} \leftarrow (y, (y_i)_{i=1}^n)$
for $i \leftarrow 1 \ldots t$ **do**
  $\tau \leftarrow \mathcal{A}$
  $\mathcal{A} \leftarrow x_\tau$
end for
if IND-TCCA1 or IND-TCCA2 **then**
  while $(c, \tau) \leftarrow \mathcal{A}$ **do** $\mathcal{A} \leftarrow \mathsf{DecShare}(x_\tau, c)$ **end while**
end if
$(m_0, m_1) \leftarrow \mathcal{A}$
$b \xleftarrow{R} \{0, 1\}$
$c^* \leftarrow \mathsf{Enc}(y, m_b)$
$\mathcal{A} \leftarrow c^*$
if IND-TCCA2 **then**
  while $(c, \tau) \leftarrow \mathcal{A}$ **do**
    if $c = c^*$ **then** $\mathcal{A} \leftarrow \perp$ **else** $\mathcal{A} \leftarrow \mathsf{DecShare}(x_\tau, c)$ **end if**
  end while
end if
$b' \leftarrow \mathcal{A}$
return $b = b'$

E.4 Homomorphic Threshold ElGamal

Given a cyclic group G with generator g, the following is the threshold homomorphic ElGamal encryption scheme.

Setup Create a cyclic group G of prime order q (where q is approximately the size of the security parameter) and a generator g.

GenShare Pick $x \xleftarrow{R} \mathbb{Z}_q$ and compute $y = g^x$ in G.

CombineKey Given a list of public key shares $(y_i)_{i \in I}$, compute $y = \prod_{i \in I} y_i$.

Enc Given a message m, pick a random $r \xleftarrow{R} \mathbb{Z}_q$ and compute $a = g^r$ and $b = g^m y^r$. The ciphertext is $c = (a, b)$.

DecShare Given $c = (a, b)$ and x, compute $d = a^x$.

Combine Given $c = (a, b), (x_i)_{i \in I}$ compute $h = b/(\prod_{i \in I} x_i)$ which will be g^m if everything was done correctly. Use a discrete-logarithm finding algorithm (for small m) to extract m.

Add Given $a_1 = g^{r_1}, b_1 = g^{m_1} y^{r_1}$ and $a_2 = g^{r_2}, b_2 = g^{m_2} y^{r_2}$, compute $a = a_1 \cdot a_2, b = b_1 \cdot b_2$.

Election verifiability in electronic voting protocols* **

Steve Kremer[1], Mark Ryan[2], and Ben Smyth[2,3]

[1] LSV, ENS Cachan & CNRS & INRIA, France
[2] School of Computer Science, University of Birmingham, UK
[3] École Normale Supérieure & CNRS & INRIA, France

Abstract. We present a formal, symbolic definition of election verifiability for electronic voting protocols in the context of the applied pi calculus. Our definition is given in terms of boolean tests which can be performed on the data produced by an election. The definition distinguishes three aspects of verifiability: individual, universal and eligibility verifiability. It also allows us to determine precisely which aspects of the system's hardware and software must be trusted for the purpose of election verifiability. In contrast with earlier work our definition is compatible with a large class of electronic voting schemes, including those based on blind signatures, homomorphic encryption and mixnets. We demonstrate the applicability of our formalism by analysing three protocols: FOO, Helios 2.0, and Civitas (the latter two have been deployed).

1 Introduction

Electronic voting systems are being introduced, or trialled, in several countries to provide more efficient voting procedures. However, the security of electronic elections has been seriously questioned [9, 20, 8, 24]. A major difference with traditional paper based elections is the lack of transparency. In paper elections it is often possible to observe the whole process from ballot casting to tallying, and to rely on robustness characteristics of the physical world (such as the impossibility of altering the markings on a paper ballot sealed inside a locked ballot box). By comparison, it is not possible to observe the electronic operations performed on data. Computer systems may alter voting records in a way that cannot be detected by either voters or election observers. A voting terminal's software might be infected by malware which could change the entered vote, or even execute a completely different protocol than the one expected. The situation can be described as *voting on Satan's computer*, analogously with [5].

The concept of *election* or *end-to-end verifiability* that has emerged in the academic literature, *e.g.*, [17, 18, 10, 3, 21, 2], aims to address this problem. It should allow voters and election observers to verify, independently of the hardware and software running the election, that votes have been recorded, tallied and declared correctly. One generally distinguishes two aspects of verifiability.

* This work has been partly supported by the EPSRC projects *UbiVal* (EP/D076625/2), *Trustworthy Voting Systems* (EP/G02684X/1) and *Verifying Interoperability Requirements in Pervasive Systems* (EP/F033540/1); the *ANR SeSur AVOTÉ* project; and the *Direction Générale pour l'Armement* (DGA).

** A long version containing full proofs is available in [19].

- *Individual verifiability:* a voter can check that her own ballot is included in the election's bulletin board.
- *Universal verifiability:* anyone can check that the election outcome corresponds to the ballots published on the bulletin board.

We identify another aspect that is sometimes included in universal verifiability.

- *Eligibility verifiability:* anyone can check that each vote in the election outcome was cast by a registered voter and there is at most one vote per voter.

We explicitly distinguish eligibility verifiability as a distinct property.

Our contribution. We present a definition of election verifiability which captures the three desirable aspects. We model voting protocols in the applied pi calculus and formalise verifiability as a triple of boolean tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$ which are required to satisfy several conditions on all possible executions of the protocol. Φ^{IV} is intended to be checked by the individual voter who instantiates the test with her private information (*e.g.*, her vote and data derived during the execution of the protocol) and the public information available on the bulletin board. Φ^{UV} and Φ^{EV} can be checked by any external observer and only rely on public information, *i.e.*, the contents of the bulletin board.

The consideration of eligibility verifiability is particularly interesting as it provides an assurance that the election outcome corresponds to votes legitimately cast and hence provides a mechanism to detect ballot stuffing. We note that this property has been largely neglected in previous work and our earlier work [22] only provided limited scope for.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. As it is not reasonable to assume voting systems behave correctly we only model the parts of the protocol that we need to trust for the purpose of verifiability; all the remaining parts of the system will be controlled by the adversarial environment. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter*. In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and protocols often need to trust some parts of the voting software or some administrators. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or can be executed in a distributed manner amongst several different election officials. For instance, in Helios 2.0 [3], the ballot construction can be audited using a cast-or-audit mechanism. Whether trust assumptions are reasonable depends on the context of the given election, but our work makes them explicit.

Tests Φ^{IV}, Φ^{UV} and Φ^{EV} are assumed to be verified in a trusted environment (if a test is checked by malicious software that always evaluates the test to hold, it is useless). However, the verification of these tests, unlike the election, can be repeated on different machines, using different software, provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed in [2, Chapter 5].

We apply our definition on three case studies: the protocol by Fujioka *et al.* [15]; the Helios 2.0 protocol [4] which was effectively used in recent university elections in Belgium; and the protocol by Juels *et al.* [18], which has been implemented by Clarkson *et al.* as Civitas [13, 12]. This demonstrates that our definition is suitable for a large class of protocols; including schemes based on mixnets, homomorphic encryption and blind signatures. (In contrast, our preliminary work presented in [22] only considers blind signature schemes.) We also notice that Helios 2.0 does not guarantee eligibility verifiability and is vulnerable to ballot stuffing by dishonest administrators.

Related work. Juels *et al.* [17, 18] present a definition of universal verifiability in the provable security model. Their definition assumes voting protocols produce non-interactive zero-knowledge proofs demonstrating the correctness of tallying. Here we consider definitions in a symbolic model. Universal verifiability was also studied by Chevallier-Mames *et al.* [11]. They show an incompatibility result: protocols cannot satisfy verifiability and vote privacy in an unconditional way (without relying on computational assumptions). But as witnessed by [17, 18], weaker versions of these properties can hold simultaneously. Our case studies demonstrate that our definition allows privacy and verifiability to coexist (see [14, 6] for a study of privacy properties in the applied pi calculus). Baskar *et al.* [7] and subsequently Talbi *et al.* [23] formalised individual and universal verifiability for the protocol by Fujioka *et al.* [15]. Their definitions are tightly coupled to that particular protocol and cannot easily be generalised. Moreover, their definitions characterise individual executions as verifiable or not; such properties should be considered with respect to every execution.

In our earlier work [22] a preliminary definition of election verifiability was presented with support for automated reasoning. However, that definition is too strong to hold on protocols such as [18, 4]. In particular, our earlier definition was only illustrated on a simplified version of [18] which did not satisfy coercion-resistance because we omitted the mixnets. Hence, this is the first general, symbolic definition which can be used to show verifiability for many important protocols, such as the ones studied in this paper.

2 Applied pi calculus

The calculus assumes an infinite set of names $a, b, c, k, m, n, s, t, \ldots$, an infinite set of variables $v, x, y, z, \ldots$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. We also assume an infinite set of *record variables* $r, r_1, \ldots$. A function symbol of arity 0 is a constant. We use metavariables u, w to range over both names and variables. Terms L, M, N, T, U, V are built by applying function symbols to names, variables and other terms. Tuples $u_1, \ldots, u_l$ and $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{u}$ and $\tilde{M}$. We write $\{{}^{M_1}/_{x_1}, \ldots, {}^{M_l}/_{x_l}\}$ for substitutions that replace variables $x_1, \ldots, x_l$ with terms $M_1, \ldots, M_l$.

The applied pi calculus [1, **?**] relies on a simple sort system. Terms can be of sort Channel for channel names or Base for the payload sent out on these channels. Function symbols can only be applied to, and return, terms of sort Base. A term is ground when it does not contain variables.

The grammar for processes is shown in Figure 1 where u is either a name or variable of channel sort. Plain processes are standard constructs, except for the *record message* $\mathsf{rec}(r, M).P$ construct discussed below. Extended processes introduce *active substitutions* which generalise the classical let construct: the process $\nu\, x.(\{^M/_x\} \mid P)$ corresponds exactly to the process let $x = M$ in P. As usual names and variables have scopes which are delimited by restrictions and by inputs. All substitutions are assumed to be cycle-free.

$P, Q, R ::=$	processes	$A, B, C ::=$	extended processes
0	null process	P	plain process
$P \mid Q$	parallel	$A \mid B$	parallel composition
$!P$	replication	$\nu\, n.A$	name restriction
$\nu\, n.P$	name restriction	$\nu\, x.A$	variable restriction
$u(x).P$	message input	$\{^M/_x\}$	active substitution
$\overline{u}\langle M\rangle.P$	message output		
$\mathsf{rec}(r, M).P$	record message		
if $M = N$ then P else Q	conditional		

Fig. 1. Applied pi calculus grammar

A *frame* φ is an extended process built from 0 and active substitutions $\{^M/_x\}$; which are composed by parallel composition and restriction. The *domain* of a frame φ is the set of variables that φ exports. Every extended process A can be mapped to a frame $\phi(A)$ by replacing every plain process in A with 0.

The record message construct $\mathsf{rec}(r, M).P$ introduces the possibility to enter special entries in frames. We suppose that the sort system ensures that r is a variable of record sort, which may only be used as a first argument of the rec construct or in the domain of the frame. Moreover, we make the global assumption that a record variable has a unique occurrence in each process. Intuitively, this construct will be used to allow a voter to privately record some information which she may later use to verify the election.

The sets of free and bound names and variables in process A are denoted by $\mathrm{fn}(A)$, $\mathrm{bn}(A)$, $\mathrm{fv}(A)$, $\mathrm{bv}(A)$. Similarly, we write $\mathrm{fn}(M)$, $\mathrm{fv}(M)$ for the names and variables in term M and $\mathrm{rv}(A)$ and $\mathrm{rv}(M)$ for the set of record variables in a process and a term. An extended process A is *closed* if $\mathrm{fv}(A) = \emptyset$. A *context* $C[\_]$ is an extended process with a hole. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output.

The signature Σ is equipped with an equational theory E, that is, a finite set of equations of the form $M = N$. We define $=_E$ as the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names. In this paper we tacitly assume that all signatures and equational theories contain the function symbols $\mathsf{pair}(\cdot,\cdot)$, $\mathsf{fst}(\cdot)$, $\mathsf{snd}(\cdot)$ and equations for pairing:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \qquad \mathsf{snd}(\mathsf{pair}(x, y)) = y$$

as well as some constant $\bot$. As a convenient shortcut we then write $(\!|T_1, \ldots T_n|\!)$ for $\mathsf{pair}(T_1, \mathsf{pair}(\ldots, \mathsf{pair}(T_n, \bot)))$ and $\pi_i(T)$ for $\mathsf{fst}(\mathsf{snd}^{i-1}(T))$.

Semantics. The operational semantics of the applied pi calculus are defined with respect to the three relations: structural equivalence ($\equiv$), internal reductions ($\rightarrow$) and labelled reduction ($\xrightarrow{\alpha}$). These semantics are standard and defined in [19]. We only illustrate them on an example (Figure 2). We write $\Longrightarrow$ for $(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^*$, that is, the reflexive transitive closure of the labelled reduction.

Let $P = \nu a, b.\mathsf{rec}(r, a).\overline{c}\langle(\!|a, b|\!)\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle$. Then we have that

$$
\begin{array}{rcl}
P & \rightarrow & \nu a, b.(\overline{c}\langle(\!|a, b|\!)\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{a}/_{r}\}) \\
 & \equiv & \nu a, b.(\nu y_1.(\overline{c}\langle y\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\}) \mid \{^{a}/_{r}\}) \\
 & \xrightarrow{\nu x.\overline{c}\langle x\rangle} & \nu a, b.(c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
 & \xrightarrow{\nu x.c(\pi_1(y))} & \nu a, b.(\text{if } a = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
 & \rightarrow & \nu a, b.(\overline{c}\langle f(a)\rangle \mid \{\!| \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
 & \xrightarrow{\nu y_2.\overline{c}\langle y_2\rangle} & \nu a, b.(\text{if } a = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{f(a)}/_{y_2}\} \mid \{^{a}/_{r}\}
\end{array}
$$

Observe that labelled outputs are done by reference and extend the domain of the process's frame.

Fig. 2. A sequence of reductions in the applied pi semantics

3 Formalising voting protocols

As discussed in the introduction we want to explicitly specify the parts of the election protocol which need to be trusted. Formally the trusted parts of the voting protocol can be captured using a voting process specification.

Definition 1 (Voting process specification). *A* voting process specification *is a tuple $\langle V, A\rangle$ where V is a plain process without replication and A is a closed evaluation context such that $fv(V) = \{v\}$ and $rv(V) = \emptyset$.*

For the purposes of individual verifiability the voter may rely on some data derived during the protocol execution. We keep track of all such values using the record construct (Definition 2).

Definition 2. *Let* rv *be an infinite list of distinct* record variables*. We define the function* R *on a finite process P without replication as $\mathsf{R}(P) = \mathsf{R}_{\mathsf{rv}}(P)$ and, for all lists rv:*

$$
\begin{array}{ll}
\mathsf{R}_{rv}(0) & \widehat{=}\ 0 \\
\mathsf{R}_{rv}(P \mid Q) & \widehat{=}\ \mathsf{R}_{\mathit{odd}(rv)}(P) \mid \mathsf{R}_{\mathit{even}(rv)}(Q) \\
\mathsf{R}_{rv}(\nu\, n.P) & \widehat{=}\ \nu\, n.\mathsf{rec}(\mathit{head}(rv), n).\mathsf{R}_{\mathit{tail}(rv)}(P) \\
\mathsf{R}_{rv}(u(x).P) & \widehat{=}\ u(x).\mathsf{rec}(\mathit{head}(rv), x).\mathsf{R}_{\mathit{tail}(rv)}(P) \\
\mathsf{R}_{rv}(\overline{u}\langle M\rangle.P) & \widehat{=}\ \overline{u}\langle M\rangle.\mathsf{R}_{rv}(P) \\
\mathsf{R}_{rv}(\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q) & \widehat{=}\ \mathit{if}\ M = N\ \mathit{then}\ \mathsf{R}_{rv}(P)\ \mathit{else}\ \mathsf{R}_{rv}(Q)
\end{array}
$$

where the functions head and tail are the usual ones for lists, and odd (resp. even) returns the list of elements in odd (resp. even) position.

In the above definition odd and even are used as a convenient way to split an infinite list into two infinite lists.

Given a sequence of record variables $\tilde{r}$, we denote by $\tilde{r}_i$ the sequence of variables obtained by indexing each variable in $\tilde{r}$ with i. A voting process can now be constructed such that the voter V records the values constructed and input during execution.

Definition 3. *Given a voting process specification* $\langle V, A\rangle$*, integer* $n \in \mathbb{N}$*, and names* $s_1, \ldots, s_n$*, we build the augmented voting process* $\mathsf{VP}_n^+(s_1, \ldots, s_n) = A[V_1^+ \mid \cdots \mid V_n^+]$ *where* $V_i^+ = \mathsf{R}(V)\{^{s_i}/_v\}\{^{r_i}/_r \mid r \in rv(\mathsf{R}(V))\}$.

The process $\mathsf{VP}_n^+(s_1, \ldots, s_n)$ models the voting protocol for n voters casting votes $s_1, \ldots, s_n$, who privately record the data that may be needed for verification using record variables $\tilde{r}_i$.

4 Election verifiability

We formalise election verifiability using three tests Φ^{IV}, Φ^{UV}, Φ^{EV}. Formally, a test is built from conjunctions and disjunctions of *atomic tests* of the form $(M =_E N)$ where M, N are terms. Tests may contain variables and will need to hold on frames arising from arbitrary protocol executions. We now recall the purpose of each test and assume some naming conventions about variables.

Individual verifiability: The test Φ^{IV} allows a voter to identify her ballot in the bulletin board. The test has:

- a variable v referring to a voter's vote.
- a variable w referring to a voter's public credential.
- some variables $x, \bar{x}, \hat{x}, \ldots$ expected to refer to global public values pertaining to the election, *e.g.*, public keys belonging to election administrators.
- a variable y expected to refer to the voter's ballot on the bulletin board.
- some record variables $r_1, \ldots, r_k$ referring to the voter's private data.

Universal verifiability: The test Φ^{UV} allows an observer to check that the election outcome corresponds to the ballots in the bulletin board. The test has:

- a tuple of variables $\tilde{v} = (v_1, \ldots, v_n)$ referring to the declared outcome.
- some variables $x, \bar{x}, \hat{x}, \ldots$ as above.
- a tuple $\tilde{y} = (y_1, \ldots, y_n)$ expected to refer to all the voters' ballots on the bulletin board.
- some variables $z, \bar{z}, \hat{z}, \ldots$ expected to refer to outputs generated during the protocol used for the purposes of universal and eligibility verification.

Eligibility verifiability: The test Φ^{EV} allows an observer to check that each ballot in the bulletin board was cast by a unique registered voter. The test has:

- a tuple $\tilde{w} = (w_1, \ldots, w_n)$ referring to public credentials of eligible voters.
- a tuple $\tilde{y}$, variables $x, \bar{x}, \hat{x}, \ldots$ and variables $z, \bar{z}, \hat{z}, \ldots$ as above.

The remainder of this section will focus on the individual and universal aspects of our definition; eligibility verifiability will be discussed in Section 5.

4.1 Individual and universal verifiability

The tests suitable for the purposes of election verifiability have to satisfy certain conditions: if the tests succeed, then the data output by the election is indeed valid (*soundness*); and there is a behaviour of the election authority which produces election data satisfying the tests (*effectiveness*). Formally these requirements are captured by the definition below. We write $\tilde{T} \simeq \tilde{T'}$ to denote that the tuples $\tilde{T}$ and $\tilde{T'}$ are a permutation of each other modulo the equational theory, that is, we have $\tilde{T} = T_1, \ldots T_n$, $\tilde{T'} = T'_1, \ldots T'_n$ and there exists a permutation χ on $\{1, \ldots, n\}$ such that for all $1 \leq i \leq n$ we have $T_i =_E T'_{\chi(i)}$.

Definition 4 (Individual and universal verifiability). *A voting specification $\langle V, A\rangle$ satisfies* individual and universal verifiability *if for all $n \in \mathbb{N}$ there exist tests Φ^{IV}, Φ^{UV} such that $fn(\Phi^{IV}) = fn(\Phi^{UV}) = rv(\Phi^{UV}) = \emptyset$, $rv(\Phi^{IV}) \subseteq rv(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ the conditions below hold. Let $\tilde{r} = rv(\Phi^{IV})$ and $\Phi_i^{IV} = \Phi^{IV}\{^{s_i}/_v, ^{\tilde{r}_i}/_{\tilde{r}}\}$.*

Soundness. For all contexts C and processes B such that $C[\mathsf{VP}_n^+(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:

$$\forall i, j. \quad \Phi_i^{IV}\sigma \wedge \Phi_j^{IV}\sigma \Rightarrow i = j \tag{1}$$

$$\Phi^{UV}\sigma \wedge \Phi^{UV}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma \Rightarrow \tilde{v}\sigma \simeq \tilde{v}'\sigma \tag{2}$$

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{IV}\{^{y_i}/_y\}\sigma \wedge \Phi^{UV}\sigma \;\Rightarrow\; \tilde{s} \simeq \tilde{v}\sigma \tag{3}$$

Effectiveness. There exists a context C and a process B, such that $C[\mathsf{VP}_n^+(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{IV}\{^{y_i}/_y\}\sigma \wedge \Phi^{UV}\sigma \tag{4}$$

An individual voter should verify that the test Φ^{IV} holds when instantiated with her vote s_i, the information $\tilde{r}_i$ recorded during the execution of the protocol and some bulletin board entry. Indeed, Condition (1) ensures that the test will hold for at most one bulletin board entry. (Note that Φ_i^{IV} and Φ_j^{IV} are evaluated with the same ballot $y\sigma$ provided by $C[\_]$.) The fact that her ballot is counted will be ensured by Φ^{UV} which should also be tested by the voter. An observer will instantiate the test Φ^{UV} with the bulletin board entries $\tilde{y}$ and the declared outcome $\tilde{v}$. Condition (2) ensures the observer that Φ^{UV} only holds for a single outcome. Condition (3) ensures that if a bulletin board contains the ballots of voters who voted $s_1, \ldots, s_n$ then Φ^{UV} only holds if the declared outcome is (a permutation of) these votes. Finally, Condition (4) ensures that there exists an execution where the tests hold. In particular this allows us to verify whether the protocol can satisfy the tests when executed as expected. This also avoids tests which are always false and would make Conditions (1)-(3) vacuously hold.

4.2 Case study: FOO

The FOO protocol, by Fujioka, Okamoto & Ohta [15], is an early scheme based on blind signatures and has been influential for the design of later protocols.

How FOO works. The voter first computes her ballot as a commitment to her vote $m' = \mathsf{commit}(rnd, v)$ and sends the signed blinded ballot $\mathsf{sign}(sk_V, \mathsf{blind}(rnd', m'))$ to the registrar. The registrar checks that the signature belongs to an eligible voter and returns $\mathsf{sign}(sk_R, \mathsf{blind}(rnd', m'))$, the blind signed ballot. The voter verifies the registrar's signature and unblinds the message to recover her ballot signed by the registrar $m = \mathsf{sign}(sk_R, m')$. The voter then posts her signed ballot to the bulletin board. Once all votes have been cast the tallier verifies all the entries and appends an identifier ℓ to each valid entry. The voter then checks the bulletin board for her entry, the triple (ℓ, m', m), and appends the commitment factor rnd. Using rnd the tallier opens all of the ballots and announces the declared outcome.

Equational theory. We model blind signatures and commitment as follows.

$$\begin{aligned} \mathsf{checksign}(\mathsf{pk}(x), \mathsf{sign}(x, y)) &= \mathsf{true} & \mathsf{getmsg}(\mathsf{sign}(x, y)) &= y \\ \mathsf{unblind}(y, \mathsf{sign}(x, \mathsf{blind}(y, z))) &= \mathsf{sign}(x, z) & \mathsf{unblind}(x, \mathsf{blind}(x, y)) &= y \\ \mathsf{open}(x, \mathsf{commit}(x, y)) &= y \end{aligned}$$

Model in applied pi. The parts of the protocol that need to be trusted for achieving verifiability are surprisingly simple (Definition 5). The name rnd models the randomness that is supposed to be used to compute the commitment of the vote. All a voter needs to ensure is that the randomness used for the commitment is fresh. To ensure verifiability, all other operations such as computing the commitment, blinding and signing can be performed by the untrusted terminal.

Definition 5. *The voting process specification* $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}} \rangle$ *is defined as*

$$V_{\mathsf{foo}} \mathrel{\hat{=}} \nu rnd.\bar{c}\langle v \rangle.\bar{c}\langle rnd \rangle \quad \textit{and} \quad A_{\mathsf{foo}}[\_] \mathrel{\hat{=}} \_.$$

Individual and universal verifiability. We define the tests

$$\Phi^{IV} \mathrel{\hat{=}} y =_E (\!|r, \mathsf{commit}(r, v)|\!) \qquad \Phi^{UV} \mathrel{\hat{=}} \bigwedge_{1 \le i \le n} v_i =_E \mathsf{open}(\pi_1(y), \pi_2(y))$$

Intuitively, a bulletin board entry y should correspond to the pair formed of the random generated by voter i and commitment to her vote.

Theorem 1. $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}} \rangle$ *satisfies individual and universal verifiability.*

4.3 Case study: Helios 2.0

Helios 2.0 [4] is an open-source web-based election system, based on homomorphic tallying of encrypted votes. It allows the secret election key to be distributed amongst several trustees, and supports distributed decryption of the election result. It also allows independent verification by voters and observers of election results. Helios 2.0 was successfully used in March 2009 to elect the president of the Catholic University of Louvain, an election that had 25,000 eligible voters.

How Helios works. An election is created by naming a set of trustees and running a protocol that provides each of them with a share of the secret part of a public key pair. The public part of the key is published. Each of the eligible voters is also provided with a private pseudo-identity. The steps that participants take during a run of Helios are as follows.

1. To cast a vote, the user runs a browser script that inputs her vote and creates a ballot that is encrypted with the public key of the election. The ballot includes a ZKP that the ballot represents an allowed vote (this is needed because the ballots are never decrypted individually).
2. The user can audit the ballot to check if it really represents a vote for her chosen candidate; if she elects to do this, the script provides her with the random data used in the ballot creation. She can then independently verify that the ballot was correctly constructed, but the ballot is now invalid and she has to create another one.
3. When the voter has decided to cast her ballot, the voter's browser submits it along with her pseudo-identity to the server. The server checks the ZKPs of the ballots, and publishes them on a bulletin board.
4. Individual voters can check that their ballots appear on the bulletin board. Any observer can check that the ballots that appear on the bulletin board represent allowed votes, by checking the ZKPs.
5. The server homomorphically combines the ballots, and publishes the encrypted tally. Anyone can check that this tally is done correctly.
6. The server submits the encrypted tally to each of the trustees, and obtains their share of the decryption key for that particular ciphertext, together with a proof that the key share is well-formed. The server publishes these key shares along with the proofs. Anyone can check the proofs.
7. The server decrypts the tally and publishes the result. Anyone can check this decryption.

Equational theory. We use a signature in which $\mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}})$ denotes the encryption with key x_{pk} and random x_{rand} of the plaintext x_{text}, and $x_{\mathsf{ciph}} * y_{\mathsf{ciph}}$ denotes the homomorphic combination of ciphertexts x_{ciph} and y_{ciph} (the corresponding operation on plaintexts is written $+$ and on randoms $\circ$). The term $\mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s, x_{\mathsf{ballot}})$ represents a proof that the ballot x_{ballot} contains some name s and random x_{rand} with respect to key x_{pk}; $\mathsf{decKey}(x_{\mathsf{sk}}, x_{\mathsf{ciph}})$ is a decryption key for x_{ciph} w.r.t. public key $\mathsf{pk}(x_{\mathsf{sk}})$; and $\mathsf{decKeyPf}(x_{\mathsf{sk}}, x_{\mathsf{ciph}}, x_{\mathsf{dk}})$ is a proof that x_{dk} is a decryption key for x_{ciph} w.r.t. public key $\mathsf{pk}(x_{\mathsf{sk}})$. We use the equational theory that asserts that $+, *, \circ$ are commutative and associative, and includes the equations:

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})) = x_{\mathsf{text}}$$

$$\mathsf{dec}(\mathsf{decKey}(x_{\mathsf{sk}}, ciph), ciph) = x_{\mathsf{plain}} \quad \text{where } ciph = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$$

$$\mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}}, y_{\mathsf{text}}) * \mathsf{penc}(x_{\mathsf{pk}}, z_{\mathsf{rand}}, z_{\mathsf{text}}) = \mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}} \circ z_{\mathsf{rand}}, y_{\mathsf{text}} + z_{\mathsf{text}})$$

$$\mathsf{checkBallotPf}(x_{\mathsf{pk}}, ballot, \mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s, ballot)) = \mathsf{true} \quad \text{where } ballot = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s)$$

$$\mathsf{checkDecKeyPf}(\mathsf{pk}(x_{\mathsf{sk}}), ciph, dk, \mathsf{decKeyPf}(x_{\mathsf{sk}}, ciph, dk)) = \mathsf{true}$$
where $ciph = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$and $dk = \mathsf{decKey}(x_{\mathsf{sk}}, ciph)$

Note that in the equation for $\mathsf{checkBallotPf}$ s is a name and not a variable. As the equational theory is closed under bijective renaming of names this equation holds for any name, but fails if one replaces the name by a term, *e.g.*, $s + s$. We suppose that all names are possible votes but give the possibility to check that a voter does not include a term $s + s$ which would add a vote to the outcome.

Model in applied pi. The parts of the system that are not verifiable are:

- The script that constructs the ballot. Although the voter cannot verify it, the trust in this script is motivated by the fact that she is able to audit it.
- The trustees. Although the trustees' behaviour cannot be verified, voters and observers may want to trust them because trust is distributed among them.

Hence, we include these two components in the context A_{helios} of our voting process specification.

Definition 6. *The voting process specification* $\langle V_{\mathsf{helios}}, A_{\mathsf{helios}}\rangle$ *is defined where*

$$\begin{aligned}
V_{\mathsf{helios}} &\hat{=} d(x_{\mathsf{pid}}).\ \overline{d}\langle v\rangle.\ d(x_{\mathsf{ballot}}).\ d(x_{\mathsf{ballotpf}}).\overline{c}\langle(\!|w, x_{\mathsf{ballot}}, x_{\mathsf{ballotpf}}|\!)\rangle\\
A_{\mathsf{helios}}[_] &\hat{=} \nu sk, d.\ \left(\overline{c}\langle\mathsf{pk}(sk)\rangle \mid (!\nu pid.\ \overline{d}\langle pid\rangle) \mid (!B) \mid T \mid _\right)\\
B &\hat{=} \nu m.\ d(x_{\mathsf{vote}}).\overline{d}\langle\mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}})\rangle.\\
&\qquad \overline{d}\langle\mathsf{ballotPf}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}, \mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}))\rangle\\
T &\hat{=} c(x_{\mathsf{tally}}).\ \overline{c}\langle(\!|\mathsf{decKey}(sk, x_{\mathsf{tally}}), \mathsf{decKeyPf}(sk, x_{\mathsf{tally}}, \mathsf{decKey}(sk, x_{\mathsf{tally}}))|\!)\rangle
\end{aligned}$$

We suppose that the inputs of x_{pid}, x_{ballot} and x_{ballotpf} are stored in record variables r_{pid}, r_{ballot} and $r_{ballotpf}$ respectively. The voter V_{helios} receives her voter id pid on a private channel. She sends her vote on the channel to A_{helios}, which creates the ballot for her. She receives the ballot and sends it (paired with pid) to the server. A_{helios} represents the parts of the system that are required to be trusted. It publishes the election key and issues voter ids. It includes the ballot creation script B, which receives a voter's vote, creates a random m and forms the ballot, along with its proof, and returns it to the voter. A_{helios} also contains the trustee T, which accepts a tally ciphertext and returns a decryption key for it, along with the proof that the decryption key is correct. We assume the trustee will decrypt any ciphertext (but only one).

The untrusted server is assumed to publish the election data. We expect the frame to define the election public key as x_{pk} and the individual pid's and ballots as y_i for each voter i. It also contains the homomorphic tally z_{tally} of the encrypted ballots, and the decryption key z_{decKey} and its proof of correctness z_{decKeyPf} obtained from the trustees. When the protocol is executed as expected the resulting frame should have substitution σ such that

$$\begin{aligned}
y_i\sigma &= (\!|pid_i, \mathsf{penc}(\mathsf{pk}(sk), m_i, v_i), \mathsf{ballotPf}(\mathsf{pk}(sk), m_i, v_i, \mathsf{penc}(\mathsf{pk}(sk), m_i, v_i))|\!)\\
x_{pk}\sigma &= \mathsf{pk}(sk) \qquad\qquad z_{\mathsf{tally}}\sigma = \pi_2(y_1) * \cdots * \pi_2(y_n)\sigma\\
z_{\mathsf{decKey}}\sigma &= \mathsf{decKey}(sk, z_{\mathsf{tally}})\sigma \quad z_{\mathsf{decKeyPf}}\sigma = \mathsf{decKeyPf}(sk, z_{\mathsf{tally}}, z_{\mathsf{decKey}})\sigma
\end{aligned}$$

Individual and universal verifiability. The tests Φ^{IV} and Φ^{UV} are introduced for verifiability purposes. Accordingly, given $n \in \mathbb{N}$ we define:

$$\begin{array}{rl}\Phi^{IV} & \widehat{=}\ y =_E (\!| r_{pid}, r_{ballot}, r_{ballotpf} |\!) \\ \Phi^{UV} & \widehat{=}\ z_{\mathsf{tally}} =_E \pi_2(y_1) * \cdots * \pi_2(y_n) \\ & \quad \wedge \bigwedge_{i=1}^{n} (\mathsf{checkBallotPf}(x_{\mathsf{pk}}, \pi_2(y_i), \pi_3(y_i)) =_E \mathsf{true}) \\ & \quad \wedge \mathsf{checkDecKeyPf}(x_{\mathsf{pk}}, z_{\mathsf{tally}}, z_{\mathsf{decKey}}, z_{\mathsf{decKeyPf}}) =_E \mathsf{true} \\ & \quad \wedge v_1 + \cdots + v_n =_E \mathsf{dec}(z_{\mathsf{decKey}}, z_{\mathsf{tally}})\end{array}$$

Theorem 2. *$\langle V_{\mathsf{helios}}, A_{\mathsf{helios}} \rangle$ satisfies individual and universal verifiability.*

5 Eligibility verifiability

To fully capture *election verifiability*, the tests Φ^{IV} and Φ^{UV} must be supplemented by a test Φ^{EV} that checks eligibility of the voters whose votes have been counted. We suppose that the public credentials of eligible voters appear on the bulletin board. Φ^{EV} allows an observer to check that only these individuals (that is, those in posession of credentials) cast votes, and at most one vote each.

Definition 7 (Election verifiability). *A voting specification $\langle V, A \rangle$ satisfies* election verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$ such that $fn(\Phi^{IV}) = fn(\Phi^{UV}) = fn(\Phi^{EV}) = rv(\Phi^{UV}) = rv(\Phi^{EV}) = \emptyset$, $rv(\Phi^{IV}) \subseteq rv(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ we have:*

1. *The tests Φ^{IV} and Φ^{UV} satisfy each of the conditions of Definition 4;*
2. *The additional conditions 5, 6, 7 and 8 below hold.*

Let $\tilde{r} = rv(\Phi^{IV})$, $\Phi_i^{IV} = \Phi^{IV}\{{}^{s_i}/_{v}, {}^{\tilde{r}_i}/_{\tilde{r}}, {}^{y_i}/_{y}\}$, $X = fv(\Phi^{EV}) \backslash dom(\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n))$

Soundness. For all contexts C and processes B such that $C[\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:

$$\Phi^{EV}\sigma \wedge \Phi^{EV}\{{}^{x'}/_{x} \mid x \in X \backslash \tilde{y}\}\sigma \Rightarrow \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{5}$$

$$\bigwedge_{1 \le i \le n} \Phi_i^{IV}\sigma \wedge \Phi^{EV}\{{}^{\tilde{w}'}/_{\tilde{w}}\}\sigma \quad \Rightarrow \quad \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{6}$$

$$\Phi^{EV}\sigma \wedge \Phi^{EV}\{{}^{x'}/_{x} \mid x \in X \backslash \tilde{w}\}\sigma \Rightarrow \tilde{y}\sigma \simeq \tilde{y}'\sigma \tag{7}$$

Effectiveness. There exists a context C and a process B such that $C[\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)]$ $\Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and

$$\bigwedge_{1 \le i \le n} \Phi_i^{IV}\sigma \wedge \Phi^{UV}\sigma \wedge \Phi^{EV}\sigma \tag{8}$$

The test Φ^{EV} is instantiated by an observer with the bulletin board. Condition (5) ensures that, given a set of ballots $\tilde{y}\sigma$, provided by the environment, Φ^{EV} succeeds only for one list of voter public credentials. Condition (6) ensures that if a bulletin board contains the ballots of voters with public credentials $\tilde{w}\sigma$ then Φ^{EV} only holds on a permutation of these credentials. Condition (7) ensures that, given a set of credentials $\tilde{w}$, only one set of bulletin board entries $\tilde{y}$ are accepted by Φ^{EV} (observe that for such a strong requirement to hold we expect the voting specification's frame to contain a public key, to root trust). Finally, the effectiveness condition is similar to Condition (4) of the previous section.

5.1 Case study: JCJ-Civitas

The protocol due to Juels *et al.* [18] is based on mixnets and was implemented by Clarkson *et al.* [13, 12] as an open-source voting system called Civitas.

How JCJ-Civitas works. An election is created by naming a set of registrars and talliers. The protocol is divided into four phases: *setup*, *registration*, *voting* and *tallying*. We now detail the steps of the protocol, starting with the setup phase.

1. The registrars (resp. talliers) run a protocol which constructs a public key pair and distributes a share of the secret part amongst the registrars' (resp. talliers'). The public part $\mathsf{pk}(sk_T)$ (resp. $\mathsf{pk}(sk_R)$) of the key is published. The registrars also construct a distributed signing key pair ssk_R, $\mathsf{pk}(ssk_R)$.

The registration phase then proceeds as follows.

2. The registrars generate and distribute voter credentials: a private part d and a public part $\mathsf{penc}(\mathsf{pk}(sk_R), m'', d)$ (the probabilistic encryption of d under the registrars' public key $\mathsf{pk}(sk_R)$). This is done in a distributed manner, so that no individual registrar learns the value of any private credential d.
3. The registrars publish the signed public voter credentials.
4. The registrars announce the candidate list $\tilde{t} = (t_1, \ldots, t_l)$.

The protocol then enters the voting phase.

5. Each voter selects her vote $s \in \tilde{t}$ and computes two ciphertexts $M = \mathsf{penc}(\mathsf{pk}(sk_T), m, s)$ and $M' = \mathsf{penc}(\mathsf{pk}(sk_R), m', d)$ where m, m' are nonces. M contains her vote and M' her credential. In addition, the voter constructs a non-interactive zero-knowledge proof of knowledge demonstrating the correct construction of her ciphertexts and validity of the candidate ($s \in \tilde{t}$). (The ZKP provides protection against coercion resistance, by preventing forced abstention attacks via a *write in*, and binds the two ciphertexts for eligibility verifiability.) The voter derives her ballot as the triple consisting of her ciphertexts and zero-knowledge proof and posts it to the bulletin board.

After some predefined deadline the tallying phase commences.

6. The talliers read the n' ballots posted to the bulletin board by voters (that is, the triples consisting of the two ciphertexts and the zero-knowledge proof) and discards any entries for which the zero-knowledge proof does not hold.
7. The elimination of re-votes is performed on the ballots using pairwise *plaintext equality tests* (PET) on the ciphertexts containing private voter credentials. (A PET [16] is a cryptographic predicate which allows a keyholder to provide a proof that two ciphertexts contain the same plaintext.) Re-vote elimination is performed in a verifiable manner with respect to some publicly defined policy, *e.g.*, by the order of ballots on the bulletin board.
8. The talliers perform a verifiable re-encryption mix on the ballots (ballots consist of a vote ciphertext and a public credential ciphertext; the link between both is preserved by the mix.) The mix ensures that a voter cannot trace her vote, allowing the protocol to achieve coercion-resistance.
9. The talliers perform a verifiable re-encryption mix on the list of public credentials published by the registrars. This mix anonymises public voter credentials, breaking any link with the voter for privacy purposes.
10. Ballots based on invalid credentials are weeded using PETs between the mixed ballots and the mixed public credentials. Both have been posted to the bulletin board. (Using PETs the correctness of weeding is verifiable.)
11. Finally, the talliers perform a verifiable decryption and publish the result.

Equational theory. The protocol uses a variant of the ElGamal encryption scheme [18]. Accordingly we adopt the signature and associated equational theory from the Helios case study. We model the ZK proof demonstrating correct construction of the voter's ciphertexts, re-encryption and PETs by the equations

$$\begin{aligned}
&\mathsf{checkBallot}(\mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}, x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}}),\\
&\qquad\qquad \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}), \mathsf{penc}(x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}})) = \mathsf{true}\\
&\mathsf{renc}(y_{\mathsf{rand}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})) = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), f(x_{\mathsf{rand}}, y_{\mathsf{rand}}), x_{\mathsf{text}})\\
&\mathsf{pet}(\mathsf{petPf}(x_{\mathsf{sk}}, ciph, ciph'), ciph, ciph') = \mathsf{true}
\end{aligned}$$

where $ciph \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})$ and $ciph' \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x'_{\mathsf{rand}}, x_{\mathsf{text}})$. In addition we consider verifiable re-encryption mixnets and introduce for each permutation χ on $\{1, \ldots, n\}$ the equation:

$$\begin{aligned}
&\mathsf{checkMix}(\mathsf{mixPf}(x_{\mathsf{ciph},1}, \ldots, x_{\mathsf{ciph},n}, ciph_1, \ldots, ciph_n, z_{\mathsf{rand},1}, \ldots, z_{\mathsf{rand},n}),\\
&\qquad\qquad x_{\mathsf{ciph},1}, \ldots, x_{\mathsf{ciph},n}, ciph_1, \ldots, ciph_n) = \mathsf{true}
\end{aligned}$$

where $ciph_i \mathrel{\hat{=}} \mathsf{renc}(z_{\mathsf{rand},i}, x_{\mathsf{ciph},\chi(i)})$. We also define re-encryption of pairs of ciphertexts and introduce for each permutation χ on $\{1, \ldots, n\}$ the equation

$$\begin{aligned}
&\mathsf{checkMixPair}(\mathsf{mixPairPf}(\langle\!| x_1, x'_1 |\!\rangle, \ldots, \langle\!| x_n, x'_n |\!\rangle, \langle\!| c_1, c'_1 |\!\rangle, \ldots, \langle\!| c_n, c'_n |\!\rangle,\\
&\quad \langle\!| z_1, z'_1 |\!\rangle, \ldots, \langle\!| z_n, z'_n |\!\rangle), \langle\!| x_1, x'_1 |\!\rangle, \ldots, \langle\!| x_n, x'_n |\!\rangle, \langle\!| c_1, c'_1 |\!\rangle, \ldots, \langle\!| c_n, c'_n |\!\rangle) = \mathsf{true}
\end{aligned}$$

where $c_i \mathrel{\hat{=}} \mathsf{renc}(z_i, x_{\chi(i)})$ and $c'_i \mathrel{\hat{=}} \mathsf{renc}(z'_i, x'_{\chi(i)})$.

Model in applied pi. We make the following trust assumptions for verifiability

- The voter is able to construct her ballot; that is, she is able to generate nonces m, m', construct her ciphertexts and generate a zero-knowledge proof.
- The registrars construct distinct credentials d for each voter and construct the voter's public credential correctly. (The latter assumption can be dropped if the registrars provides a proof that the public credential is correctly formed [18].) The registrars keep the private part of the signing key secret.

Although neither voters nor observers can verify that the registrars adhere to such expectations, they trust them because trust is distributed. The trusted components are modelled by the voting process specification $\langle A_{\mathsf{jcj}}, V_{\mathsf{jcj}}\rangle$ (Definition 8). The context A_{jcj} distributes private keys on a private channel, launches an unbounded number of registrar processes and publishes the public keys of both the registrars and talliers. The registrar R constructs a fresh private credential d and sends the private credential along with the signed public part (that is, $\mathsf{sign}(ssk_R, \mathsf{penc}(x_{pk_R}, m'', d))$) to the voter; the registrar also publishes the signed public credential on the bulletin board. The voter V_{jcj} receives the private and public credentials from the registrar and constructs her ballot; that is, the pair of ciphertexts and a zero-knowledge proof demonstrating their correct construction.

Definition 8. *The voting process specification $A_{\mathsf{jcj}}, V_{\mathsf{jcj}}$ is defined where:*

$$
\begin{aligned}
A_{\mathsf{jcj}} &\mathrel{\hat{=}} \nu\, a, ssk_R.(!R \mid \{\mathsf{pk}(sk_R)/_{x_{\mathsf{pk_R}}}, \mathsf{pk}(ssk_R)/_{x_{spk_R}}, \mathsf{pk}(sk_T)/_{x_{\mathsf{pk_T}}}\} \mid \_)\\
V_{\mathsf{jcj}} &\mathrel{\hat{=}} \nu\, m, m'.a(x_{cred}).\mathit{let}\ ciph = \mathsf{penc}(x_{pk_T}, m, v)\ \mathit{in}\\
&\quad \mathit{let}\ ciph' = \mathsf{penc}(x_{pk_R}, m', \pi_1(x_{cred}))\ \mathit{in}\\
&\quad \mathit{let}\ zkp = \mathsf{ballotPf}(x_{pk_T}, m, v, x_{pk_R}, m', \pi_1(x_{cred}))\ \mathit{in}\\
&\quad \overline{c}\langle(\!|ciph, ciph', zkp|\!)\rangle\\
R &\mathrel{\hat{=}} \nu\, d, m''.\ \mathit{let}\ sig = \mathsf{sign}(ssk_R, \mathsf{penc}(x_{pk_R}, m'', d))\ \mathit{in}\ \overline{a}\langle(\!|d, sig|\!)\rangle.\overline{c}\langle sig\rangle
\end{aligned}
$$

Election verifiability. We suppose the recording function uses record variables $\tilde{r} = (r_{cred}, r_m, r_{m'}) = \mathsf{rv}(\mathsf{R}(V))$ (corresponding to the variable x_{cred} and names m, m' in the process V). Accordingly, given $n \in \mathbb{N}$ we define:

$$
\begin{aligned}
\Phi^{IV} &\mathrel{\hat{=}} y =_E (\!|\mathsf{penc}(x_{\mathsf{pk_T}}, r_m, v), \mathsf{penc}(x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred})),\\
&\qquad \mathsf{ballotPf}(x_{\mathsf{pk_T}}, r_m, v, x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred}))|\!) \wedge w = \pi_2(r_{cred})\\
\Phi^{UV} &\mathrel{\hat{=}} \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!),\\
&\qquad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i})) =_E v_i\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{checkDecKeyPf}(x_{\mathsf{pk_T}}, \pi_1(z_{\mathsf{bal},i}), z_{\mathsf{decKey},i}, z_{\mathsf{decPf},i}) =_E \mathsf{true}\\
\Phi^{EV} &\mathrel{\hat{=}} \textstyle\bigwedge_{i=1}^{n} \mathsf{checkBallot}(\pi_3(y_i), \pi_1(y_i), \pi_2(y_i))\\
&\quad \wedge \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!),\\
&\qquad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{pet}(z_{\mathsf{petPf},i}, \pi_2(z_{\mathsf{bal},i}), \hat{z}_{\mathsf{cred},i}) =_E \mathsf{true}\\
&\quad \wedge (\!|z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}|\!) \simeq (\!|\hat{z}_{\mathsf{cred},1}, \ldots, \hat{z}_{\mathsf{cred},n}|\!)\\
&\quad \wedge \mathsf{checkMix}(z_{\mathsf{mixPf}}, \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{checksign}(x_{spk_R}, w_i)
\end{aligned}
$$

Theorem 3. *$\langle A_{\mathsf{jcj}}, V_{\mathsf{jcj}}\rangle$ satisfies election verifiability.*

6 Conclusion

We present a symbolic definition of election verifiability which allows us to precisely identify which parts of a voting system need to be trusted for verifiability. The suitability of systems can then be evaluated and compared on the basis of trust assumptions. We also consider eligibility verifiability, an aspect of verifiability that is often neglected and satisfied by only a few protocols, but nonetheless an essential mechanism to detect ballot stuffing. We have applied our definition to three protocols: FOO, which uses blind signatures; Helios 2.0, which is based on homomorphic encryption, and JCJ-Civitas, which uses mixnets and anonymous credentials. For each of these protocols we discuss the trust assumptions that a voter or an observer needs to make for the protocol to be verifiable. Since Helios 2.0 and JCJ-Civitas have been implemented and deployed, we believe our formalisation is suitable for analysing real world election systems.

Acknowledgements

We are particularly grateful to Michael Clarkson for careful reading of our preliminary formal definition of election verifiability. His comments provided useful guidance for the definition we present here.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01: Proc. 28th ACM Symposium on Principles of Programming Languages*, pages 104–115, New York, USA, 2001. ACM.
2. B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.
3. B. Adida. Helios: Web-based open-audit voting. In *Proc. 17th Usenix Security Symposium*, pages 335–348. USENIX Association, 2008.
4. B. Adida, O. de Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2009.
5. R. Anderson and R. Needham. Programming Satan's Computer. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 426–440. Springer, 1995.
6. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *CSF'08: Proc. 21st IEEE Computer Security Foundations Symposium*, pages 195–209, Washington, USA, 2008. IEEE.
7. A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *TARK'07: Proc. 11th International Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71. ACM, 2007.
8. D. 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/elections/voting_systems/ttbr/db07_042_ttbr_system_decisions_release.pdf`, August 2007.

9. Bundesverfassungsgericht (Germany's Federal Constitutional Court). Use of voting computers in 2005 Bundestag election unconstitutional. Press release 19/2009 http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html, March 2009.
10. D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *ESORICS'05: Proc. 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.
11. B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traore. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Proc. Workshop on Trustworthy Elections*, 2006.
12. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. Technical Report 2007-2081, Cornell University, May 2007. Revised March 2008. http://hdl.handle.net/1813/7875.
13. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *S&P'08: Proc. Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.
14. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
15. A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *ASIACRYPT'92: Proc. Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.
16. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT'00: Proc. 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–177. Springer, 2000.
17. A. Juels, D. Catalano, and M. Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.
18. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *WPES '05: Proc. workshop on Privacy in the electronic society*, pages 61–70. ACM, 2005. See also http://www.rsa.com/rsalabs/node.asp?id=2860.
19. S. Kremer, B. Smyth, and M. D. Ryan. Election verifiability in electronic voting protocols. Technical Report CSR-10-06, University of Birmingham, School of Computer Science, 2010. Available at http://www.bensmyth.com/publications/10tech/CSR-10-06.pdf.
20. Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (Netherland's Ministry of the Interior and Kingdom Relations). Stemmen met potlood en papier (Voting with pencil and paper). Press release http://www.minbzk.nl/onderwerpen/grondwet-en/verkiezingen/nieuws--en/112441/stemmen-met-potlood, May 2008.
21. Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord. http://www.dagstuhlaccord.org/, 2007.
22. B. Smyth, M. D. Ryan, S. Kremer, and M. Kourjieh. Towards automatic analysis of election verifiability properties. In *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, LNCS. Springer, 2010. To appear.
23. M. Talbi, B. Morin, V. V. T. Tong, A. Bouhoula, and M. Mejri. Specification of Electronic Voting Protocol Properties Using ADM Logic: FOO Case Study. In *ICICS'08: Proc. 10th International Conference on Information and Communications Security Conference*, pages 403–418, London, 2008. Springer.
24. UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. http://www.electoralcommission.org.uk/elections/pilots/May2007.