Contego
Laboratories

Review of Electronic Voting Protocol Models and Proofs (Combined Final Report)

Author by
David Basin, Srdjan Čapkun

Recipient
Post CH AG, Business Area E-Voting,
Hr. Denis Morel, Wankdorfallee 4,
CH-3030 Berne

24.05.2017

Summary

This report summarizes the results of our audit of models and proofs of Swiss Post/Scytl's Electronic Voting Protocol. The audit and associated reporting was carried out over the period January – May 2017.

This summary report comprises the two reports we wrote. The first report describes the results of the first audit and the five non-conformities that we identified with respect to the requirements of the Bundeskanzelei. These issues were addressed by Post/Scytl in a substantial revision to their submitted documents. Our second report was written as an addendum to the first. As explained in this report, the improvements made resolved all the previously documented non-conformities with respect to the cryptographic and symbolic proofs. In both reports, we identified additional (minor, non-blocking) issues and improvements.

We summarize here our main conclusion: the proofs are conformant to requirements of the Bundeskanzlei (Chapter 5.1).

This audit report was produced by the undersigned on May 24th, 2017.

Prof. Dr. David Basin

Prof. Dr. Srdjan Capkun

Contego
Laboratories

Addendum to Review of Electronic Voting Protocol Models and Proofs

Final Report (v1.0)

Author by
David Basin, Srdjan Čapkun

Recipient
Post CH AG, Business Area E-Voting,
Hr. Denis Morel, Wankdorfallee 4,
CH-3030 Berne

24.05.2017

Table of Contents

1. Executive Summary

This report is an addendum to our previous report "Review of Electronic Voting Protocols, Models and Proofs (v2.0)", completed on April 8th, 2017. This addendum addresses the five sources of non-conformity summarized in Section 6 of our previous report and analyzes whether the system is still non-conformant to the requirements of the Bundeskanzelei [BK, 5.1].

We received both new design and proof documents, which we reviewed. The design document clarified which features of the system are relevant for the review. The proof documents and proof scripts are substantially improved with respect to their consistency, their explanation of the abstractions used, and the proofs themselves.

The improvements have resolved all the previously documented non-conformities with respect to the cryptographic and symbolic proofs, as discussed in Section 4 of this report. We have, however, identified some residual issues, which are discussed in the same section.

2. Received Documents and Auxiliary Documents

The documents we used for the review were the original documents we received for the first review as well as the following updated documents:

- **[OV]** Online Voting, Protocol Specifications, v4.8
- **[Crypto]** Swiss Online Voting System, Cryptographic proofs of Individual Verifiability Protocol (revision 2.0, dated April 7th, 2017)
- **[Symbolic]** Analysis of Cast-as-Intended Verifiability and Ballot Privacy Properties for Scytl's Swiss On-line Voting Protocol using ProVerif (v2, April 7th, 2017)
- **[ProVerif Sources]** Source files containing the symbolic model, including the formalization of the adversary capabilities and security properties. Namely, the following 22 files:
    - equiv_v6.7-Check.pve, equiv_v6.7-DHH.pve, equiv_v6.7-HDH.pve, equiv_v6.7-HH.pve, equiv_v6.7-HHD.pve, spec_v4.3-Check.pve, spec_v4.3-noEq.pve, spec_v4.3.pve, spec_v8-Check.pve, spec_v8-noEq.pve, spec_v8.pve, study_v10.pve, study_v11-Check.pve, study_v11.pve, study_v12-Check.pve, study_v12.pve, study_v13-OBS_HH_k=1.pv study_v13-OBS_k=1-Check.pv study_v13-OBS_k=1.pv study_v13-REA-Check.pve, study_v13-REA.pve, study_v13.pve.

We will refer as well to other documents cited in our previous report, in particular:

- **[BK]** Technical and administrative requirements for electronic vote casting, Swiss Bundeskanzelei, v.1.00.

When there is a possible ambiguity as to which version of a document is referenced, we include the document's version number. For example, [OV, v4.7] refers to the design document from the previous review.

3. Objectives and Evaluation Scope

This report is the result of a second evaluation whose main objective is to evaluate the five previously identified non-conformities, with respect to the updated design and proofs.

Below we recall the nonconformities identified in Section 6 of our previous report.

NC1. Design documents should describe the complete design of the Swiss implementation. These documents should explicitly describe all options (e.g., extended authentication, receipts, etc.) that will be used in the final system. Also, all control flow should be made explicit in the design documents, including how errors and exceptional cases are handled.

NC2. Relationship of models to the design.
This should be made explicit so that the correspondence between the models and the design is clear. Keys, data, procedures, and protocols phases and steps should all be identically named across the different documents: [OV], [Crypto], [Symbolic], and [Proverif Sources]. When models omit or modify details from the design, this should also be stated explicitly and justified.

NC3. Provide missing details in models and proofs.
Models and proofs should be provided for write-in votes (cryptographic and symbolic), Maurer's Unified Proofs (cryptographic), and the use of short codes versus long codes (cryptographic and symbolic). The proofs should be amended to reflect the design in [OV].

NC4. Game hopping proofs.
These require additional details and minor changes, e.g., the reduction from Game A1 to A2, additions to allow adaptive attacks, and more information on how the NIZK proofs are handled.

NC5. Symbolic Proofs.
The models should be extended to formalize the relationship between different keys. Moreover, missing control flow details, which are present in the design, should be added. In cases where such details are inadequately explained in the design documentation, for example [OV, §4.4], then [OV] should also be extended. Additional explanation is also required on the modeling of NIZK proofs and why it is adequate for the different kinds of NIZK proofs actually used in the design.

We note that the first two points concern the design and its relationship to the model, whereas the last three points address the proofs themselves.

4. Evaluation and Recommendations

a. Evaluation Methodology

We carried out again the evaluation methodology described in our original report. Namely, to validate the proofs we carried out two types of checks, the first is the relationship of the model to the design and the second is the correctness of the proofs. We also checked whether the changes introduced new non-conformities and, in particular, whether they resolve the main issues identified in our previous report.

b. Evaluation of Previous Non-conformities and Recommendations

We evaluate below the five non-conformities in light of the revisions

NC1. **Design documents should describe the complete design of the Swiss implementation.**

Status: We received a new design document that clarified the previously identified design issues. In particular:

- The design document [OV, pg.8] clarifies that it is for a system where write-in votes are not allowed. This resolves the issue present in [OV, v4.7] that write-in votes were part of the design, but not reflected in the model.
- Appendix C was added to [OV] describing the extended authentication option used in the Canton Fribourg. In this extension, users must authenticate themselves by answering a challenge question in addition to entering a voting key. (Note: user authentication was omitted from both models.)
- Additional details are provided on receipt validation, which differs between cantons [OV, Section 4.8].

Remaining issues:

- Although the revised documents [OV] and [Crypto] now use the same or similar notation and are much more consistent than they previously were, each of the documents includes a separate description of the voting protocol. [OV] covers the whole protocol specification, and [Crypto] provides more details on the cryptographic primitives but covers only the subset of operations that are mainly relevant for the cast-as-intended property.
  In [Crypto] Section 1.1. it is stated "In this paper we present the protocol of what we call the Swiss Online Voting System, which is the platform developed by Swiss Post and Scytl that is used in several Swiss Cantons.". [Crypto] does not refer to the *Online Voting Protocol Specifications V4.8* [OV] document that is (according to our understanding) the main protocol specification document.
  We recommend create one document that merges these two descriptions to avoid any confusion and to improve readability.
- [OV] and [Crypto] do not fully describe the control flow of different systems. We illustrate this via an example of ProcessVote(). As stated in [Crypto], this function

first "verifies that there is not already a vote in BB for the Voting Card ID." This requires further elaboration. Does this mean that ProcessVote() would output 0 if the same function was already invoked in the past for that ID? Or would it output 0 only if the vote for this ID was already confirmed by the user and sVCC was already produced? This relates to the question of whether the voters are allowed to change their votes. This is a part of the functional specification and should be made explicit both in the protocol description and in the models.
As we noted in our original report "all control flow should be made explicit in the design documents, including how errors and exceptional cases are handled". We still recommend that this is done.

NC2. Relationship of models to the design.

Status: The revision resolved this non-conformity. [Crypto, Appendix A] now explains the abstractions that are made in the model and the relationship to the Swiss Online Voting System, especially when it comes to entities, identities, keys and authentication procedures. However, there are still two issues that we recommend addressing:

- Both [Crypto] and [Symbolic] should clearly refer to particular sections in [OV] to clarify the correspondence to the design document. This is analogous to the traceability of requirements in system development.
- The naming across documents is much improved, but more care is still needed. Here are some representative examples:
    - CMtable in CreateRC [Symbolic] corresponds to CM_id in [Crypto]
    - Function AliceData in [Symbolic] corresponds to part of Register in [Crypto]
    - Procedure CreateRC in [Symbolic] corresponds (roughly) to CreateCC in [Crypto].
    - "Alice" in [Symbolic] corresponds to "voter" in [Crypto]
    - $J_1, ..., j_k$ for voting choices in [Symbolic] versus $v_1, ..., v_t$ in [Crypto]
    - Voting server in [Symbolic] corresponds to Bulletin Board Manager in [Crypto]
    - ProcessVoteCheck in [Symbolic] corresponds to ProcessVote in [Crypto]

  Again, consistent naming is needed for design traceability.

NC3. Provide missing details in models and proofs.

Status: We received new proof documents for both the cryptographic [Crypto] and the symbolic [Symbolic] proofs. As noted before, write-in votes have been eliminated from the system design, and hence need no longer be part of the model. Both documents were updated to explain how zero-knowledge proofs are modeled. Moreover, short and long choice codes were added to both models and proofs.

NC4. Game hopping proofs.

Status: Game hooping proofs now include sufficient additional details regarding the use of NIZKPs and proof steps. The revision therefore addressed the non-conformities.

NC5. Symbolic Proofs.

Status: The description of the symbolic models [Symbolic] and the symbolic models themselves [ProVerif Sources] were substantially improved. This includes:

- The model was expanded with relevant keys from [OV]. The model thereby included relevant voter keys, electoral board keys, Vote Cast Code keys, voter password, Verification Card Keys, and Ballot Casting Keys.
- Both long Vote Cast Codes and short Vote Cast Codes were modeled as was their relationship.
- The account of non-interactive zero knowledge proofs was simplified, explanations were added explaining the relationship to the algorithm CreateVote in [OV], and a discussion was included of limitations stemming from the tool used.
- The changes in [Symbolic] were consistently made in the new ProvVerif models, e.g., study_v13-REA.pve. Moreover, minor comments on the symbolic proof, made in the original report, were taken into account in the revised models.

c. Summary of Recommendations

We summarize below our main recommendations:

1. We recommend accepting the improvements as resolving the previously identified five non-conformities.

2. In the case of the first two non-conformities, NC1 and NC2, there are four remaining issues, identified above, that we still recommend to be addressed. Their essence is that a single, unambiguous, description of the e-voting protocols should be created, with an explicit control flow. This description should be clearly used as the source of the cryptographic and symbolic models, and the relationship between these artifacts should be documented in a way that removes all ambiguity about how the models were derived, the abstractions that were used, and the parts of the design that were omitted from the models. Given the current state of the models and proofs, we do not view these as blocking issues that should delay the system's certification. However, we do recommend that they are nevertheless addressed.

5. Conclusions

Given the improvements made to the design, proof documents, and proof scripts, we view the system as conformant to the requirements of the Bundeskanzelei [BK, 5.1]. The proofs themselves are well done and representative of state-of-the-art verification methods.

Nevertheless, we have listed issues—both in this addendum and the original report—which we believe should be addressed. The most serious of which is extending the design document to make all control flow in the protocol explicit and explicitly linking the modeled protocols to the protocol described in [OV], i.e., supporting the traceability of the [OV] protocol design in the cryptographic and symbolic models.

Finally, we note that establishing universal verifiability at some later time will be more complex. At that point, if not before, the recommendations made here for individual verifiability will need to be addressed.

Responsibility

This audit report was produced by the undersigned on May 24th, 2017.

Prof. Dr. David Basin

Prof. Dr. Srdjan Capkun

Contego
Laboratories

Review of Electronic Voting Protocol Models and Proofs
Final Report (v2.0)

Author by
David Basin, Srdjan Čapkun

Recipient
Post CH AG, Business Area E-Voting,
Hr. Denis Morel, Wankdorfallee 4,
CH-3030 Berne

08.04.2017

Table of Contents

1. Executive Summary

We reviewed the symbolic and cryptographic proofs of the Swiss Electronic Voting Protocols, focusing primarily on whether the protocols ensure individual verifiability. We summarize our findings here.

Both the cryptographic and symbolic proofs were constructed using well-established methods and tools, widely accepted within the cryptographic and verification communities. The cryptographic proofs use the game hopping approach, building on computational assumptions. The symbolic proofs follow the symbolic approach to modeling and verification and are checked using the ProVerif tool, which is a state-of-the-art model-checking tool. The proofs were constructed by experts in these domains.

Although the models and proofs are well done, we have identified some issues that should be addressed before they can be fully accepted. First, the protocol models should be amended so that the correspondence between the design (documents [OV] and [Tools]) and the abstractions used in the models is made clear. Currently, the proof documents do not refer to the design documents and also use different terminology than those in the design documents. Second, in both the cryptographic and symbolic models, some relevant steps (the use of short codes, write-in votes, voter authentication) have been omitted. Specifically, short codes should be included in the models (and resulting proofs) since they are at the core of cast-as-intended verifiability and therefore of individual verifiability. Third, in terms of the proofs themselves: some extensions and additional details should be added to the cryptographic proofs. This includes refining how Noninteractive Zero Knowledge proofs are handled, refining the hops between games, formalizing the relationship between different keys, adding missing control flow details, etc.

A complication we faced is that the design documents we received do not contain the complete details how the design will be customized for Swiss elections. For example, it is unclear which particular options (e.g., extended authentication, receipts, etc.) will be used in the final system. Some of this can be inferred from public sources [PostWeb], but it is essential to have one document describing the design of the Swiss Online Voting System with all the options and choices worked out.

Finally, there are discrepancies in the security properties considered across different documents. The cryptographic proof document proves cast-as-intended verifiability and vote privacy; the symbolic proof document also proves tallied-as-cast verifiability. Federal requirements [BK] require the proof of individual verifiability, which from the definition given in the requirement includes cast-as-intended verifiability, but not recorded-as-cast. However, some literature [E2E] as well as the Post website on electronic voting [PostWeb] define individual verifiability as cast-as-intended and recorded-as-cast; the latter is neither covered by the cryptographic nor the symbolic proof. We have taken the narrow interpretation of the properties (just cast-as-intended verifiability), in this report. Nevertheless, these discrepancies should be resolved to determine whether the scope of the proofs should be extended to include recorded-as-cast.

2. Received Documents and Auxiliary Documents

For the review, we received a number of documents from Scytl and Swiss Post. We list them below along with the abbreviations we will reference them by. The main **design documents** that we received are:

- **[OV]** Online Voting, Protocol Specifications, v4.7
- **[Tools]** Online Voting, Cryptographic Tools Specification, v1.1

The **verification documents** that we received are:

- **[Crypto]** Swiss Online Voting Protocol (revision 1.2, dated Feb. 22nd, 2017)
- **[Symbolic]** Analysis of Cast-as-Intended Verifiability and Ballot Privacy Properties for Scytl's Swiss On-line Voting Protocol using ProVerif (No version, dated Nov. 30th, 2016)
- **[ProVerif Sources]** Source files containing the symbolic model, including the formalization of the adversary capabilities and security properties. Namely:
    - spec_v4.3-noEq.pve, spec_v4.3-Check.pve, equiv_v6.7-DHH.pve, equiv_v6.7-HH.pve, equiv_v6.7-HDD.pve, spec_v8-Check.pve, spec_v8-noEq.pve, spec_v8.pve, spec_v4.3.pve, equiv_v6.7-Check.pve

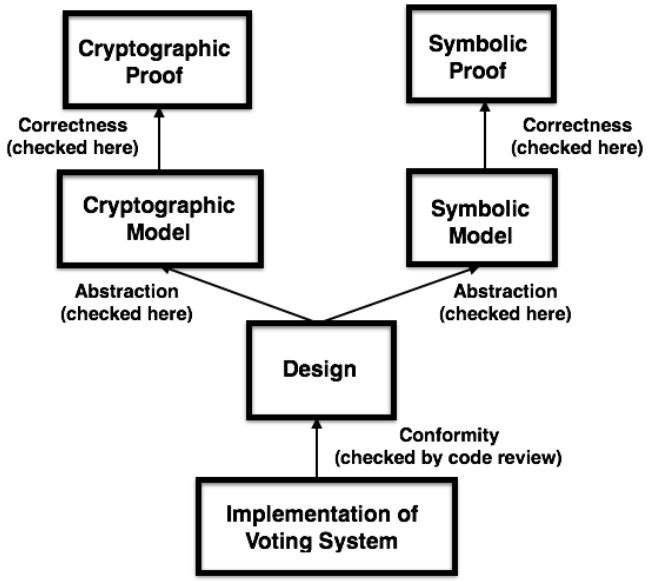
Additional documents explicitly provided by Scytl and Post are:

- **[OVW]** Crypto Proofs Doc, Extended Information, v0.1
- **[Arch]** Online Voting, Architecture Overview, v2.0
- Online Voting Solution (Scytl slides, no version number)
- Online Voting: Voter Portal, Secure Data Manager, and Administrator Portal (all v6)
- eVoting Security State of the Art Presentation for KPMG (Scytl slides)

Additional documents, in the public domain, which we reference in our evaluation, are:

- **[BK]** Technical and administrative requirements for electronic vote casting, Swiss Bundeskanzelei, v.1.00
- **[PostWeb]** Papers and slides publically available online at *https://www.post.ch/en/business/a-z-of-subjects/industry-solutions/swiss-post-e-voting*
- **[E2E]** End-to-End Verifiability, Benaloh, Rivest, Ryan, Stark, Teague, Vora, Overseas Vote Foundation's End-to-End Verifiable Internet Voting: Specification and Feasibility Assessment Study (E2E VIV Project), *http://research.microsoft.com/en-us/um/people/benaloh/papers/e2e-primer.pdf*

3. Objectives and Evaluation Scope



To clarify the scope of this evaluation, we depict the relevant entities above as well as their relationships. Our evaluation starts with the design, described primarily by the design documents [OV] and [Tools]. Out of scope is the system's implementation and whether the implementation conforms to the given design.

The main security property we evaluate is individual verifiability. To determine whether the system has this property, as required by [BK, §5.1], we must evaluate two proofs: a cryptographic proof and a symbolic proof, which are made with respect to a cryptographic model and a symbolic model of the design, respectively. In each case, this evaluation requires:

- Checking that the model is an adequate cryptographic (respectively symbolic) model of the design. Namely, that the protocol modeled appropriately abstracts the voting protocol described in the design document. For the cryptographic model, this is the protocol described by the algorithms in [Crypto]. For the symbolic model, this is the protocol described by the [Proverif Sources] we received, which are explained in the [Symbolic] paper.
- Checking that the properties verified cryptographically (or symbolically) are sufficient to ensure individual verifiability. This also includes checking that the trust assumptions (i.e., adversary model) correspond to those in [BK], i.e., the trust assumptions are no stronger than those in [BK], or equivalently the adversary is at least as strong as the one in [BK].
- Verifying the correctness of the proofs themselves. In the case of the cryptographic proof, this amounts to checking a mathematical argument. For the symbolic proofs, we check them using ProVerif, and rely on the soundness of this model checker.

We have structured our evaluation along the tasks just described.

Remarks on Individual Verifiability

The main property relevant for this review is **individual verifiability**. For individual verifiability to hold, [BK, §4.1] requires that the adversarial actions of changing or misappropriating a vote before its registration, or casting a vote on behalf of a voter will be detected with high likelihood. A stronger property, **end-to-end** (also known as **universal**) **verifiability** is typically broken down into three parts [E2E]:

1. *Cast-as-intended:* voters make their selections and, at the time of vote casting, can get convincing evidence that their encrypted votes accurately reflect their choices;
2. *Recorded-as-cast:* voters or their designees can check that their encrypted votes have been correctly included, by finding exactly the encrypted value they cast on a public list of encrypted cast votes; and
3. *Tallied-as-recorded:* any member of the public can check that all the published encrypted votes are correctly included in the tally, without knowing how any individual voted.

Individual verifiability is often understood as (1) and (2), whereas (3) is additionally required for the stronger property of universal verifiability. Note that individual verifiability as required by [BK, §4.1] is entailed by (1), provided voters make appropriate checks. Note too that the document *Process to secure & verifiable online voting* [PostWeb] provides the same breakdown as [E2E]: the voting system achieves the **cast-as-intended** property by using return codes that voters can check against their voting cards and it achieves **recorded-as-cast** by providing receipts that voters can check against a public bulletin board. Note that the document describing the cryptographic proofs [Crypto] explicitly focuses on cast-as-intended only and does not cover recorded-as-cast. The situation is similar for [Symbolic], although additional properties are also established, as explained in Section 5. **Hence, our primary focus is just on the cast-as-intended property.**

Note that vote secrecy is **not** a formal requirement for individual verifiability. It is only required in [BK, §4.3] if one wishes to verify the *complete abstract model* with the intent of showing universal verifiability. Hence this property is not in the scope of our evaluation.

Remark on Report Structure: Mandatory Versus Recommended Improvements

All improvements that we consider to be mandatory for the system to be compliant to the requirements of [BK, 5.1] are listed in Section 6, which summarizes our main recommendations. The other chapters and the appendix provide both additional information on the issues identified as well as other issues and recommendations of a more minor (i.e., non-mandatory) nature.

4. Evaluation of the Cryptographic Model and Proofs

a. Methodology

The report [Crypto] describes cryptographic protocols for electronic voting, and provides cryptographic proofs for the given protocols. The cryptographic proofs follow the well-established methodology of proving the correctness of protocols schemes using game hoping proofs in the computational model. This methodology is appropriate for the type of protocols that are analyzed.

Validating the proofs requires two types of checks, the first is the relationship to the design and the second is the correctness of the proofs. We now consider each of these in turn.

b. Evaluation of the Model (relationship to the design)

Overall, the presented model covers most relevant aspects of the voting protocol design, specifically those relevant for the cast-as-intended verifiability. We do, however, identify the following issues.

Main issues:

1. Relationship to the design. The relationship between the model (protocol description) presented in the cryptographic analysis document [Crypto] and the specification document [OV] is not explicitly given. The [Crypto] document does not refer to the main design document [OV] and the keys, data, and procedures in the [Crypto] document do not generally correspond to those in the [OV] document. Only a coarse correspondence between different steps in [OV] and [Crypto] has been provided in [OVW]. Most importantly, there is no explanation in the [Crypto] document about which steps, keys and data are abstracted away and why.
2. The use of short codes versus long codes. In [Crypto], the model and the proofs only consider the system using long return codes (called Choice Codes in [OV]). The [OV] document, however, explains the procedure for the generation and verification of short choice codes (sCC in [OV]), which substantially differs from what is described in the [Crypto]. Short string comparison protocols are non-trivial and should be handled with care. This is important since cast-as-intended is guaranteed by the use of choice codes. The analysis of the use of short choice codes therefore should be included in the cryptographic proofs and not handled informally as it is currently done in [Crypto, §5.2]. Similar to the case of Choice Codes, the confirmation codes and finalization codes (named Ballot Cast Key and Vote Cast Code in [OV]) are generated differently in [OV] than as described in [Crypto]. The proofs should be amended to reflect the design in [OV].
3. Write-in votes. The [OV] document describes the procedure for write-in votes. If write-in votes will be used, then the [Crypto] report should be extended to include this. It is our understanding from discussions with project participants that the system should be evaluated without considering the option of write-in votes, hence we do not list this as a source of non-conformance in Section 6.

4. NIZK Proofs. The cryptographic proofs describe individual Noninteractive Zero Knowledge (NIZK) proofs while the cryptographic tool specification document [Tools] uses Maurer's Unified Proofs and the instantiations of NIZK proofs within Maurer's scheme. There should be a statement about the relationship between the two in the [Crypto] report. It is clear that Maurer's protocol is a generalization, but the report should show how their proofs also carry over to Maurer's generalization and instantiations in [Tools].

Additional issues:

5. Cryptographic Keys. There are a number of keys listed in the [OV] document: Voting Card Signing and Authentication keys, Verification Card ID Key, Vote Cast Code Signer Key, Vote Card Secret Key, Choice Codes Public Key, etc. The [Crypto] document lists a smaller set of keys without providing any mapping to the [OV] document and using different terminology and notation (voters voting keys, global audit key, etc.). We were able to establish this mapping and found this simplification to be acceptable for proving cast-as-intended. However, there should be justification for this simplification in the report. We note that this simplification might not carry over when other properties are proven. We further recommend unifying the notations in the [OV] and [Crypto] documents to support traceability.
6. Scope. [PostWeb] describes receipts that will be used by the voters to check if their vote was recorded-as-cast. As is, [Crypto] does not model receipts and does not prove recorded-as-cast. This is clearly stated in the [Crypto] document. It therefore does not prove individual verifiability according to the definition from [E2E] and [PostWeb], but can be understood as fulfilling the individual verifiability requirements given in [BK, §4.1], where individual verifiability is defined differently. If recorded-as-cast is required, then the report should be extended to prove this.
7. Privacy. [Crypto] provides a proof of vote privacy. However, it does not model authentication aspects as described in [OV]. It instead addresses these informally in 5.1. In [Crypto], it is simply assumed that sk_id is given to the client by the voter. However, in [OV] there is a more complex process through which sk_id (VCSK in [OV]) is communicated to the client. Although we find this to be sufficient for cast-as-intended, for other properties, including the privacy property, which is analyzed in the report, this is not sufficient. Ideally, the user's authentication to the server and the delivery of the credentials to the client should be a part of the model and proofs. The [OV] document further describes extended authentication options. If these are to be used in the Swiss voting system, then this should be addressed in the report. Summarizing, we find this acceptable for the proofs of cast-as-intended verifiability, but not comprehensive enough for vote privacy.
8. Voter Behavior. This issue is more a side remark, but one that we believe is important for the security of the system in practice. Commonly, the behavior of voters is not captured in the analysis of cryptographic protocols. However, the presented protocols heavily rely on voters performing appropriate checks and following the protocol faithfully. Although it is not required by [BK], we nevertheless recommend making explicit what is assumed about voter behavior and that possible social engineering attacks (e.g., by a compromised client), could lead to attacks. The proofs could also be extended to capture the resilience

of the system under failures of (some subset of) voters to perform some of the checks or steps. Since the voting client is assumed to be under the adversary's control, it has extensive possibilities for manipulations.

c. Evaluation of the Proofs

[Crypto, §4] structures the proof of cast-as-intended in terms of three games. In essence, these cover the probabilities that the adversary obtains return codes (RC) corresponding to the voter's choice without this vote being cast, and that it obtains the confirmation codes (CC) or finalization codes (FC) for the voter whose choice it is trying to compromise, assuming that all cryptographic primitives, including those for the NIZK proofs, can be broken only with a negligible probability. The fourth game is used to prove vote privacy.

Overall, the proofs are well executed and they cover the relevant cases (with respect to RC, CC and FC) that could impact cast-as-intended. Assuming that the voters are trustworthy and follow the protocol, these games, by-and-large, prove cast-as-intended verifiability for the protocols that are described in [Crypto] (note that as discussed in the previous section, these protocols are an abstraction of the design [OV]). Here we discuss main issues that should be addressed.

Main issues

9. Game hopping. The game hopping proofs omit some relevant details. For example, the reduction steps in Games A and B are not included in the report. In particular, the reduction from Game A1 to A2 should be explained in more details. It should be clarified too how PerfectProcessBallot checks that "the NIZK proofs have been correctly generated". According to our understanding of the proof, PerfectProcessBallot *never* fails in checking the NIZK proofs; it is unclear how this is implemented and whether this can be done in polynomial time. This polynomial time requirement is important for the reduction since PerfectProcessBallot is called by the distinguisher algorithm in Game A2.
10. Random voter ID. For vote integrity (cast-as-intended), the adversary is assigned a random voter that he has to attack. This modeling omits the (realistic) threat of adaptive attacks in which the adversary chooses the voter to attack after having analyzed the voters' public cryptographic material. The formalization and proofs should be extended to cover adaptive adversaries.
11. NIZK proofs. The proof (Game A) that the attacker cannot cast unintended votes without being detected omits too many details and should be expanded to explain the use and relevance of individual NIZK proofs for cast-as-intended verifiability. This will also provide more clarity in terms of which hardness assumptions are relevant for cast-as-intended.

Additional Issues are described in the appendix.

5. Evaluation of the Symbolic Model and Proofs

a. Methodology

The verification methodology used corresponds to the state-of-the-art for cryptographic protocol verification using symbolic methods. We start by briefly reviewing this methodology. Rather than working in the computational cryptographic setting, where one reasons about bit strings, probabilistic polynomial time, and the adversary's advantage, this setting is abstracted to one where:

1. Agents running a protocol are modeled as communicating processes. Protocol messages are modeled by terms in a term algebra and cryptographic operators are modeled by equations or reduction rules.
2. The adversary is also defined by a process (or a set of rules) that model the knowledge that he can derive from messages he has constructed or seen, or publicly available knowledge. Together, instances of the processes defining the protocol agents and the adversary are composed, and this comprises a (global) transition system.
3. Properties such as authentication and agreement are formulated as reachability properties or trace properties of the transition system. Properties such as privacy are formulated in terms of the observational equivalence of pairs of processes.
4. One uses either theorem proving or algorithmic model-checking techniques to construct proofs. The state-of-the-art tools can, in the best case, establish that the given protocol has the stated properties even when an unbounded number of protocol sessions execute in parallel, that is, when arbitrarily many agents (including corrupted or adversarial ones) are running the protocol in different roles (e.g., client, server, etc.).

For the symbolic proofs, the ProVerif system was used for all four of these steps. Namely, the protocol, adversary model, and properties (Steps 1-3), were formalized in ProVerif's input language. The ProVerif system was then used to construct correctness proofs (Step 4) or provide counterexamples to incorrect theorems. What remains then is to evaluate Steps (1-3) above. This evaluation requires two kinds of checks:

1. The abstract symbolic protocol description, given in [Symbolic, §2-§3], must correspond to actual design [OV] as well as the adversary and properties required in [BK, §4.1].
2. The formalization of this protocol description, adversary, and properties must be suitably formalized in the input language of ProVerif. Note that models, like computer programs, are not unique: there are many ways to model a protocol, the adversary, and the properties. A suitable symbolic model is one that captures the relevant artifacts precisely enough that it captures all relevant attacks present in the design and hence correctness ensures that these attacks are not possible on an implementation that conforms to the design.

We describe each of these in the subsequent subsections.

b. Evaluation of the Model (relationship to the design)

The symbolic model is given by specifications in [ProVerif Sources] and is also partially described by the document [Symbolic]. We received 11 source files (ending in **.pve**) containing models written in a language that extends the input language of ProVerif to support parameterized specifications. A script is provided (called **expand**) that compiles these source files, along with parameter values, to ProVerif input files (ending in **.pv**).

The source files formalize (with minor differences) the same protocol and adversary model, but differ in the properties they establish. There are 3 classes of source files:

1. **Verifiability properties for a fixed number of voting options:** The source **spec_v4.3.pve** specifies that the voting protocol satisfies the cast-as-intended property and a form of the tallied-as-cast property. These properties are parameterized and established for a fixed number *n* of voting options, where the voter must choose exactly *k* of those options (e.g., *k = 2*, *n=3*). This is described in [Symbolic, §4].
2. **Ballot privacy properties:** These formalize that no one can learn information about the voting options of an honest user other than what can be learned from the election results alone. These properties are parameterized by the number of chosen voting options *k*. This is specified in the source files whose names begin with **equiv_v6.7** and described in [Symbolic, §5].
3. **Verifiability properties for an unbounded number of voting options:** These specify verifiability properties like in (1), but now for an unbounded number of voting options. This is achieved by using an appropriate encoding based on lists so that the specifications no longer require the parameter *n* describing the number of voting options. The specification is still parameterized in the number *k* of chosen voting options. The model is given by the source file **equiv_v8.pve** and described in [Symbolic, §5].

A number of source files are included that are minor variants of those named above, which are not relevant for this evaluation, or they formalize additional properties that constitute basic sanity checks on the modeled protocol. For example, the property in model **spec_v4.3-Check.pve** states that using the protocol (as modeled) a voter can vote and confirm her ballot. Such checks are sensible since all security properties proven are *safety properties* and will hold trivially if the model does not allow obliged behaviors, e.g., that voters can successfully vote.

The models themselves are carefully constructed and comparable to state-of-the-art symbolic models found in the security and cryptographic literature. The symbolic protocol model closely follows to the protocol description given in [Crypto] in modeling the actions taken by the different parties (constructing cryptographic message, storing and communicating data, etc.). However, the symbolic model, as explained above, represents the cryptographic operators as symbolic abstractions given by terms rather than as mathematical functions. This again corresponds to the state-of-the-art.

Below we focus on two central questions. First, whether the symbolic abstraction used omits so much detail that the model no longer captures realistic attacks on the actual implementation that may be possible by exploiting properties of the cryptographic operators which have been

abstracted away. This includes the question of whether the adversary's capabilities correspond to those required by [BK, §4.1]. The second question is whether the properties verified entail individual verifiability.

Adequacy of the protocol and adversary models.

- As explained in [Symbolic, p. 2], the symbolic models are based on the cryptographic model [Crypto]. Consequently, the symbolic models have the same modeling limitations identified for the cryptographic model, described in Section 4b of this report. In particular, the relationship to the actual design documents [OV] is not given, substantial parts of the design (e.g., user authentication) are omitted, and different names and terminology are used (e.g., for keys and algorithms). We emphasize some additional aspects of this below as well.
- The entire setup, describing the generation and relationship of cryptographic keys is missing. These relationships can be easily modeled in the symbolic setting and should not be abstracted away. Relationships between keys can be a source of attacks in practice, especially when components end up under adversary control, thereby compromising some of the keys. Symbolic model checkers are good at finding attacks in such scenarios.
- Control flow options in the design are sometimes abstracted away. These can, and should, be handled in the symbolic setting. An example of this is [OV, §4.4] stating that "*the system allows the voter to log out after sending the vote, and then log in back in order to see the choice codes again and confirm her vote. In this case, after authentication, the corresponding vote is obtained from the ballot box and steps 7-8 of* ***Send a Vote*** *are repeated.*" We note that (1) this possibility is completely omitted from the symbolic model. (2) The control flow is not further described in [OV, §4.1], and to this extent [OV] itself does not provide a complete description of the design. Finally (3), the details here might actually matter for the voting system's security. For example, without design details, it is unclear whether the adversary, in possession of the client's credentials, could repeatedly log in and out and change the voter's selection after seeing return codes.
- ProVerif's input language is typed and the typing in [ProVerif Sources] associates all cryptographic keys with asymmetric key pairs. In doing so, it abstracts away the distinction between keys used for ElGamal, RSA, and those produced, e.g., by a password-based key derivation function [Tools]. All encryption in [ProVerif Sources] is modeled with the same function, **Enc**, which returns a bitstring (representing a cipher text), rather than a pair as in [Crypto], containing the public key of the encrypting party and the cipher text. We recommend (non-mandatory) modeling these functions and their types more faithfully.
- The abstraction of modular exponentiation and zero knowledge should be better justified. This is a delicate point. Finer, more detailed models (e.g., formalizing ElGamal encryption in terms of a cyclic group) would more faithfully capture possible computations by a (polynomial bounded) attacker, but would lead to increased proof search. In some cases, by using alternative tools (Maude NPA, Tamarin, or even a higher-order logic based theorem prover) it would have been possible to give more detailed models, but at the possible cost of higher theorem proving effort. In particular, Non-

Interactive Zero Knowledge Proofs are modeled in [Symbolic] by adding just one equation (subsequently approximated in the source files by four equations) that expresses the multiplicative properties of the function Phi that aggregates votes and how it interacts with an abstraction of modular exponentiation. More justification is required for why this is adequate to model the different kinds of NIZK proofs used in the protocol as described in [OV] and in [Tools], where NIZK proofs build on Mauer's unified proofs.

Adequacy of the formalized properties.

The formalized properties have some limitations that are either inherent in the state-of-the-art or the ProVerif tool being used. These include:

- Specifications and properties are parameterized, so the guarantees proven only hold for the parameter instances considered. Moreover, due to the large sizes of the state spaces involved, the protocol can only be analyzed for small parameter values. For example, for the parameterized verifiability properties (spec_v4.3.pve) it takes roughly 7 hours for ProVerif to verify the stated properties for k=2 and n=3, i.e., for 3 ballot options of which voters must choose 2.
- The properties must be proven under additional assumptions about the behavior of honest voters and the system [Symbolic, p. 10]. These assumptions arising from technical artifacts of ProVerif's modeling formalism and its operational semantics. In particular, as explained in [Symbolic, p. 10], ProVerif does not support the synchronization of concurrent processes, and this leads to attacks that could be eliminated in practice with proper synchronization.

Although stronger guarantees would be desirable, all state-of-the-art verification methods will have analogous limitations.

The properties formalized are adequate for individual verifiability in the sense of [BK, §4,1] but not in the stronger sense of [E2E] or [PostWeb]. Specifically, two verifiability properties are actually checked, referred to in [Symbolic, §4] as Cast-as-Intended and Tallied-as-Cast. In [Symbolic], cast-as-intended formalizes that after an honest voter receives and validates the expected return codes, then the ballot box stores her ballot b. The protocol model formalizes here one honest voter playing against an unbounded number of other dishonest voters, and with a dishonest (client) computer and an honest server. This fulfills the trust assumptions of [BK, §4.1] and individual verifiability in the sense stated there. The Tallied-as-Cast property in [Symbolic] expresses additional guarantees that hold when the voter receives the finalization code from the server, in particular the ballot b satisfies the properties required for it to be accepted for tallying. This property can be seen as strengthening the cast-as-intended property with additional guarantees. However, it does not formalize the notion of Recorded-as-Cast from [E2E] or [PostWeb] as neither the model nor the property formalize the notion of the voter receiving a receipt and checking the receipt on a bulletin board. Moreover, it does not formalize Tallied-as-recorded either as it does not model the tallying phase and subsequent checks.

c. Evaluation of the Proofs

The correctness of the resulting proofs relies on the correctness (i.e., the soundness) of ProVerif. As ProVerif is one of the state-of-the art security protocol model checkers and is widely used, it is reasonable to assume this. The [ProVerif Sources] were provided for the evaluation along with the ProVerif tool (version 1.94). We reproduced the verification results by running the tool on the source files provided, thereby having the tool construct symbolic proofs for different parameter instances.

6. Summary of Main Recommendations

In the following, we summarize our most important recommendations from Sections 4, 5, and the Appendix of this report. **These changes are those that we consider to be mandatory for conformance to the requirements of [BK, 5.1].**

1. **Design documents should describe the complete design of the Swiss implementation.** These documents should explicitly describe all options (e.g., extended authentication, receipts, etc.) that will be used in the final system. Also, all control flow should be made explicit in the design documents, including how errors and exceptional cases are handled.
2. **Relationship of models to the design.** This should be made explicit so that the correspondence between the models and the design is clear. Keys, data, procedures, and protocols phases and steps should all be identically named across the different documents: [OV], [Crypto], [Symbolic], and [Proverif Sources]. When models omit or modify details from the design, this should also be stated explicitly and justified.
3. **Provide missing details in models and proofs.** Models and proofs should be provided for write-in votes (cryptographic and symbolic), Maurer's Unified Proofs (cryptographic), and the use of short codes versus long codes (cryptographic and symbolic). The proofs should be amended to reflect the design in [OV].
4. **Game hopping proofs**. These require additional details and minor changes, e.g., the reduction from Game A1 to A2, additions to allow adaptive attacks, and more information on how the NIZK proofs are handled.
5. **Symbolic Proofs**. The models should be extended to formalize the relationship between different keys. Moreover, missing control flow details, which are present in the design, should be added. In cases where such details are inadequately explained in the design documentation, for example [OV, §4.4], then [OV] should also be extended. Additional explanation is also required on the modeling of NIZK proofs and why it is adequate for the different kinds of NIZK proofs actually used in the design.

Given the above, the system as it stands does not conform to the requirements of [BK, 5.1] ("Nicht-Konformität – Verbesserung obligatorisch).

7. Appendix

a. Additional Comments on the Cryptographic Proofs

- The definition of cast-as-intended introduced at the beginning of §4.1 is somewhat misleading as it refers to the probability of the voter not noticing or detecting some events. But the proofs do not discuss such probabilities. Instead they assume that voters follow the protocol and never deviate. This should be clarified.
- The game transitions fail if the key material of a corrupted voter happens to overlap with the key material of the challenge voter. The probability of this happening is not considered in the proofs of games A–C. Such clash probabilities should be discussed and added to the bounds on the advantages. As the challenge voter and its key material is generated at the start, independent of the adversary, this probability is negligible as the number of random samples is polynomial. Overall, Theorem 1 should still hold.
- The challenge id in the games A–C is a random value fixed at the start of the game, before the cryptographic material (keys, confirmation codes, etc.) is generated. The sampling is irrelevant because the encoding of the ID never contributes to anything. So, one might as well fix some challenge voter id0.
  The approach taken rules out adaptive attacks, which are relevant in practice: The adversary can choose voters whose public keys look easy to break. Fixing one particular voter at the start deprives the adversary from this choice. The formalization should therefore start with a game in which the adversary can adaptively compromise voters and chose the challenge voter ID. Another transition may be needed to eliminate the adaptive choices.
  Adaptive choices also affect the reasoning about clashes among the cryptographic keys. With a fixed challenge voter, random choices for corrupted voters only have to avoid the corresponding random choice of the challenge voter. With adaptive choices, the random choices must avoid all previous random choices. With polynomial bounds on the number of voters and voting options, the probability should still be negligible, but the analysis of the games becomes harder.
- p. 10, configuration phase: “the election authorities set up the public parameters of the election such as […] the set of valid votes (combinations of voting options) V”. The protocol as described only supports votes where all voting options can be combined arbitrarily because the algorithm CreateRC checks in its second step that the hashed return codes of the chosen votes are on the bulletin board for the current voter, so this is just a membership test.
  The paper should explain how the protocol (and CreateRC in particular) must be modified so that only certain combinations of voting options are allowed. Explicitly enumerating all possible combinations as separate voting options leads to an exponential blowup and therefore breaks all the reduction proofs because the enumerations must be posted on the bulletin board as hashes of return codes and the posting is part of the reduction, which must run in polynomial time. Conversely, if the combinations are described compactly in terms of the hashes of the return codes, then ballot privacy is at risk. As the hashed return codes get posted on the bulletin board, the adversary might match the

allowed combinations against the encoding of the combinations and therefore infer the meaning of the hashes of the return codes.

- The hash function used in the signature scheme must be different from the hash function H used in the rest of the protocol. This assumption is missing. Otherwise, the transition from game B to game B.1 breaks because only some invocations of the hash function (those used to compute the signatures) are replaced by a random oracle, but not all (e.g., those used for hashing the return codes). That is, the ROM assumption does not apply. Clarification should be added that in Game B.1 it is only the hash function inside the signature scheme that is being replaced by a random oracle, but not the hash function H used in the rest of the protocol.
- In Game C.2, the PRF fska is replaced by two different random functions Rrc and Rfc. The two functions Rrc and Rfc should be the same. Otherwise, the adversary may distinguish games C.1 and C.2 by observing inconsistent mappings. (There is no argument that all return codes are different from the space of confirmation messages, so clashes are possible.) Moreover, there is no need for two separate functions Rrc and Rfc in the proof.
- p. 5f, on voting options and the PRF family: There is no discussion on how the number of voting options influences the choice of the security parameter; this is also missing in the implementation section. Moreover, [18] does not formally prove that exponentiation with a small prime forms a PRF. The authors should cite (or provide) a cryptographic proof, state the required hardness assumptions on the group, and make sure that their chosen group meets these assumptions.
- The reduction in Lemma 1 misses several details that are needed to produce game D.4. For example:
    - Configuration phase: Game D.4 does not publish sk_a and sk_c on the BB, but passes them directly to the adversary. Also, it initializes the sets ID, ID_c, and ID_h, which the reduction omits.
    - Registration phase: When registering a corrupt voter, game D.4 provides the voting options v_i and the return codes RC_i, whereas the reduction provides only the return codes RC_i. When processing Challenger_VoteLR queries, the reduction forgets to check the validity of votes and the ID of the voter with respect to the sets ID_c and ID_h.
    - Counting phase: Game D.4 re-checks all votes on the BB using ProcessBallot, but Tally in C' does no such checks. This requires an explanation of why the checks can be omitted.

  The report should spell out the reduction in pseudo-code-like the definitions of the games. It will then be easier to see whether the reduction really covers every detail of the game D.4. For every point that the reduction and the game differ in syntactically, there should be an explanation of why the difference is insignificant.

Minor issues:

- p. 3, RSA-PSS, last but one line: What is σ? Should this be ψ?
- p. 14, Challenger_Vote algorithm: The "= 1" in the first line should be dropped. There is already a test on the result afterwards. Similar in Games B and C.

- p. 15, line 1: The challenge ID **id** has been fixed at the configuration phase, so there is no point in letting it range over all IDs in IDh. In games A–C, there will only ever be one ID in IDh, namely the one chosen in the configuration phase.
- The names of algorithms in the different games A–C are re-used although their definitions are not the same. For example, **RndRegister** in game A is different from **RndRegister** in game B. It would improve clarity if different names were used for different algorithms in different games, even if the steps are conceptually the same.

b. Additional Comments on the Symbolic Proofs

The description in [Symbolic] differs in intended and presumably unintended ways from [ProVerif Sources]. The following are several examples of this:

- **Enc** is declared as a binary function in [Symbolic, p.2] but it is a ternary function in [ProVerif Sources].
- The attacker deduction rule for **w** [Symbolic, p.4] is not needed: from the message **m** the adversary can derive **m** without the modular exponentiation. This rule is not present either in [ProvVerif Sources].
- The description of **cut** [Symbolic, p.4] and its typing are unclear. It is described as projecting the 2nd element from a 5-tuple output by the algorithm **CreateVote**. In [Symbolic], **CreateVote** returns a tuple with at least 5 components (dependent on the number of voting choices) and in [Crypto] it returns a 3 tuple. In [Proverif Sources], **CreateVote** returns a 5 tuple and **cut** itself takes a single argument of type **bitstring.**

8. Responsibility

This audit report was produced by the undersigned on April 8th, 2017.

Prof. Dr. David Basin

Prof. Dr. Srdjan Capkun