Review of Symbolic Proofs for Swiss Post's Voting System

David Basin, Contego Laboratories
23, Nov. 2021

Overview and Main Findings

We reviewed the symbolic proofs for the cryptographic protocol used by the Swiss Post Voting System. This entailed checking that the models provided appropriately formalize the protocol, its threat model, and its required properties and also checking that the properties are provable using the cryptographic protocol verifier ProVerif. In this way we establish that a suitable abstraction of the protocol satisfies the requirements for both individual and universal verifiability and vote privacy.

Our main findings are:

1. The models (which encompass the voting protocol, trust assumptions and properties), are well made and the proofs can be successfully checked using the ProVerif verification tool.
2. The models and their proofs fulfill the requirements of the Federal Chancellery for a symbolic proof of the Swiss Post Voting Protocol.
3. The models are a substantial abstraction of the actual design (which itself is an abstraction of the actual system). We highlight places where more details could have been given and the provisos under which the results should be interpreted.

Documents used for Review

We used the publicly available documents from https://gitlab.com/swisspost-evoting/e-voting, downloaded on November 4th, 2021. Particularly relevant was the subdirectory **Symbolic-models**, which contained 4 models containing proofs of (individual and universal) verifiability properties and 2 models for verifying privacy properties. Also relevant was the subdirectory **System,** which contained the *Swiss Post Voting System, System Specification* v.0.9.7 (which we henceforth call *SysSpec*), and the *Swiss Post Voting System Architecture Document*, v0.9.1.

Guiding this review are the documents describing requirements for e-voting system evaluation available from the web pages of the Federal Chancellery (henceforth *FC*). The primary document is the *Federal Chancellery Ordinance on Electronic Voting* (OEV, Draft of 28 April, 2021). This document states verifiability requirements as well as cryptographic protocol requirements and detailed trust assumptions. Additional explanation and interpretation is provided by the document *Partial revision of the Ordinance on Political Rights and total revision of the Federal Chancellery Ordinance on Electronic Voting* (Draft of 28 April, 2021). Finally *Audit Concept (v1.3) for examining Swiss internet voting systems* provides guidance on the rights and duties of examiners.

Review Methodology and Scope

We carefully checked all of the ProVerif models against the requirements listed in Chapter 2 of the annex of the draft OEV. This includes: that the model is structured in a way that clarifies the abstract system players and abstract communication channels; that the goals formulated concerning verifiability, secrecy, and authentication are correct; that the goals are proven with respect to the appropriate assumptions on the trustworthiness of the system players and the communication channels; and that the model itself is formalized at a suitable level of abstraction, capturing all relevant aspects.

We ran ProVerif v2.03 on all of the models, checking that all claimed theorems actually hold. Our scope was limited to the analysis of the symbolic models provided. Note that a computational proof has been given for the same protocol. That was not subject to review here.

What is Modeled and Proven

There were three types of ProVerif models created for the voting protocol. They are models for (i) individual verifiability, (ii) universal verifiability, and (iii) privacy. Different versions of the models were created differing on parameters such as the number of control code components, vote options, and trust assumptions. Below we provide a summary.

Individual Verifiability Models

Two models are given for checking the properties associated with individual verifiability. Namely that the voter is given evidence to confirm that the attacker has not altered any votes cast or maliciously cast a vote on behalf of a voter that has been registered by the system. The two models differ in their parameters:

- One models a system consisting of 2 CCRs (return code control components), 2 CCMs (mixing control components), 4 voting options for voters to select from, with voters selecting 1 of the 4 voting options. The model incorporates the trust assumptions that one of the CCRs (*CCR1*) and one voter (*Alice*) are trustworthy, whereas the other CCR (*CCR2*), the voting server, the voting client, and the 2 mixing components (*CCM1* and *CCM2*) are untrustworthy. Arbitrarily many untrustworthy voters may also use the system.
- The other models a system identical to the above, except that voters select 2 of the 4 voting options.

The properties proven establish *individual verifiability*. If a voter completes her role in the protocol, then a ballot containing her intended vote has been inserted and confirmed in the ballot box. Hence if the voting choices fail to match her intention, then the voting server has cheated and this will be detected.

The models formalize the protocol as a process specified in the applied-pi calculus. The protocol is formalized as the parallel composition of processes that: initiate the election, setup the corrupted voting server, setup the dishonest agents, instantiate a role for the honest agent (*Alice*) and her corrupted device, and setup *CCR1* (*CCR2*, which is dishonest, is setup statically). The trust assumptions match those required by the Federal Chancellery, although not

all of the required number of control components are modelled.. The voting model focuses on the honest agent (*Alice*) casting her vote, receiving her short choice return code, checking it, casting her ballot casting key, and checking her vote cast return code. This formalization encompasses the two round voting protocol described in Figure 2 of SysSpec. It also provides an abstract formalization of the auxiliary protocols SendVote and ConfirmVote described in Section 5 of SysSpec. Note that an additional ProVerif model is also given that shows that the level of detail in the model is sufficient to identify the attack found by Haines and reported by Swiss Post on their gitlab.

Universal Verifiability Model
This model establishes the property of Universal Verifiability. Namely that the auditors receive a proof that confirms that after the votes were registered, the attacker did not change or insert any votes into the system.

The ProVerif model formalizes a scenario where there are 2 CCRs, 2 CCMs, and 4 voting options where voters select 2 of 4 options. The trust assumptions are that 2 voters are trustworthy, there are arbitrarily many untrustworthy voters, and of the 2 CCRs and 2 CCMs (respectively), 1 of each is trustworthy and 1 of each is untrustworthy. Moreover, the setup component is honest and the voting server and voting devices are corrupted.

The model reuses much of the model for individual verifiability. However it now also includes a formalization of the tallying phase, described in Section 6 of SysSpec. This includes modeling the mix components and their partial decryption/re-encryption performed during vote shuffling, as well as the proofs they produce, and the cleansing process The universal verifiability property is then established by showing that if an audit is successful then the mix control components (the two CCMs) have correctly performed the mixing and decryptions.

Privacy Models
The models establish the security objective of vote privacy. Namely, the attacker cannot breach voting secrecy or establish premature results unless he can control the voters or their devices. As is standard using symbolic models, vote privacy is formalized as the observational equivalence of two processes; technically this is done using ProVerif's support for bi-processes, where one bi-process description effectively describes the two processes, which should be proven to be observationally equivalent. The observational equivalence property formalized captures that the attacker cannot observe any difference between scenarios when Alice votes for option 1 and Bob votes for option 2, from the case when the votes are swapped, i.e., Alice votes for option 2 and Bob for option 1.

In contrast to the verifiability models, there are different trust assumptions. Namely, the model considers a trustworthy voting client (since if the client is untrustworthy, privacy trivially fails). There is also no explicit auditor role. Moreover, only one dishonest voter is considered. Finally, the two models provided differ in which CCMs they consider to be trustworthy (the first CCM versus one of the remaining three).

General Comments on the Models
The specifications are in the language of the applied-pi calculus, which is the input language of ProVerif. ProVerif itself is a state-of-the-art model checker for verifying symbolic models of cryptographic protocols and is well suited for carrying out this kind of analysis. The models were carried out by experts in symbolic modeling and the use of ProVerif.

The models are in general of high quality. They capture the intended operations, at a high-level of abstraction (see assumptions and limitations below) and also express the intended properties. It is unfortunate, however, that the models for individual verifiability, universal verifiability, and privacy are rather different. In particular, the model for privacy uses different naming (e.g., for keys) and abstractions (e.g., for Zero Knowledge Proofs), different initializations, and the models themselves are structured differently. This greatly complicates checking the models.

Assumptions and Limitations

As mentioned in the overview, the models and proofs fulfill the requirements of the Federal Chancellery for a symbolic proof of the Swiss Post Voting Protocol. However, they represent a substantial abstraction of the actual design. We highlight here places where more details could have been given and also explain provisos under which the results should be interpreted.

1. As previously observed there are different versions of the models created and verified, that differ on their parameters. No justification is given for the suitability of the parameters chosen, although in practice the choices are guided by the complexity of the model that the verification tool used (here ProVerif) can handle.
   We give two examples. First, ballots are considered with only 4 options. In practice there may be many more options on a ballot. Second, the ProVerif models for verifiability consider only groups of two control components (both for the return code components and the mixing components). In contrast, the Federal Chancellery requires that for the groups of redundant control components, there are four components per group. The effect of both limitations is that there are no guarantees for model instances (and therefore for concrete systems) with larger parameter values, e.g., more voting options or more control components. Note that this kind of limitation is fairly standard when using automated reasoning tools. It could be lifted in the first case (ballot options) by proving theorems for all (suitable) parameter instances; however, this would most likely require manual proof-by-induction and would be time consuming. In the second case (number of control components), the constant 2 used for the size of each group should simply be increased to 4.
2. As is to be expected in a symbolic model, cryptographic operations are modeled at a high level of abstraction and also as being “perfect”, e.g., no information is leaked, randomly generated values are unguessable, etc. . Hence the verification neither considers low-level problems in the design and implementation of the cryptographic primitives nor the possibility that the adversary could break them, e.g., using quantum computers.

3. User login, abortion, and resumption are omitted from the symbolic model. SysSpec says for example (pg. 10): *If the voter aborts the process after sending or confirming the vote, she can resume the process (potentially on a different voting device) by logging in again.* This is omitted in the ProVerif model since user login is completely omitted there and, curiously, from SysSpec as well.
4. The FC's abstract model refers to the Print component that is not present in the ProVerif model. In fact, the ProVerif model only has 4 communication channels, whereas the FC's model speaks of 10 channels. The trust assumptions for these channels in the ProVerif model are however the required ones.
5. Models are limited to voters selecting pre-defined voting options. Hence the possibility of write-in votes is not considered.
6. Voting secrecy, as proven, entails that premature election results are not released provided that the electoral authorities perform the final decryption of the votes after the election event has ended. The safeguards, both operational and technical, that should prevent this are not part of the symbolic models.
7. For privacy, in contrast to individual and universal verifiability, only one dishonest voter is considered. This leverages a result by Arapinis et al. that, in some contexts, proving privacy for three voters is sufficient. We have neither checked that result nor whether the assumptions needed to establish it hold for the given protocol. (These assumptions were not discussed in the model or its documentation, other than observing that they exist.)
8. Various model abstractions *were* carefully documented for the ProVerif models. Examples include:
    - In SysSpec, the voting client encrypts the partial choice return codes with the Choice Return Codes Encryption public key, and the control components jointly decrypt them. The symbolic model omits the encryption of the partial Choice Return Codes and strengthens the adversary since the voting client sends the partial choice return codes in plaintext.
    - Whereas the actual implementation splits the electoral board secret key among different electoral board members, the symbolic model assumes a single mixing control component that may be malicious.
    - User authentication is omitted. This means that the voter is assumed in the model to simply know the start voting key (SVK) that allows him to open the Verification Card Keystore (VCksid). This abstraction is supported by omitting authentication and assuming that the adversary knows all verification card Keystores, whereby the adversary cannot open the Keystores without learning the start voting key from the voter.

    The documentation provided is helpful as it simplifies checking the suitability of the abstractions taken. Nevertheless, the end result is that part of the proof is no longer machine checked. This could have been avoided with a more direct modeling where these details were not omitted.
9. Not all failure modes are considered in the model. That is, when some checks fail, the process (modeling part of the voting protocol) simply blocks. In some cases this reflects a lack of clarity about failure cases and control flow in SysSpec itself (e.g., consider the account of cleansing given there).

10. The correspondence between steps taken in the model and in SysSpec is often omitted and must be worked out by the reviewer. This is complicated by omissions, abstractions, renaming, and refactorings. We give two examples of this. First, the SetupVoting algorithm from SysSpec (pg. 23) isn't directly represented in the models. E.g., in the verifiability models parts of the initialization are done globally (in Part *2. Adversary capabilities and functions*) and part in a setup component process. Second, the voting phase in SysSpec is split into two sub-protocols SendVote and ConfirmVote. These could have been, but are not, represented by equivalently named processes in the ProVerif models.

This last point is important and worth expanding on. We emphasize in this regard that these deviations do ***not***, per se, represent modeling deficiencies that would render the proofs inadmissible. When doing symbolic proofs it is often necessary to introduce abstractions or deviations due to the modeling language or tool used. However, the deviations do complicate establishing a correspondence between the model and the design.

Consider one concrete example of this, namely the data *Alice* retrieves from the setup component when starting her voting process. In SysSpec she receives VCard, which is a 4-tuple containing $(SVK_{id}, \mathbf{CC}_{id}, BCK_{id}, VCC_{id})$, consisting of a starting vote key, short-choice return code, ballot casting key, and short vote cast return code. In the ProvVerif model she instead receives a 6-tuple *(id_Alice, deltaId(id_Alice), deltaPassword(id_Alice), bck(id_Alice), VCC(deltaId(id_Alice)), CC(deltaId(id_Alice),v(J1)))*. Here, one can check that all four components are present in the model. This check could be made easier for the reviewer with more consistent naming, e.g. the function *deltaPassword* that assigns the *SVK* to *Alice* is not part of SysSpec. Moreover, some work is required to understand the purpose of the additional components, which represent a minor refactoring. For example, the 6th component is the short choice return code for the choice *ja1* (the four choices are named *ja1*, …, *ja4*), which is the hardwired voting choice made by the honest voter *Alice*. (This code is computed during the setup in SysSec, but as part of the return code mapping table, computed by the algorithm *GenCMTable*.) Overall, even when these differences have a justification, as they do in this example, working them out for oneself greatly increases the work required to establish a correspondence between the different parts of the ProVerif model with those in the system specification. We return to this point below in our recommendations.

Recommendations for Future Reviews

When using verification tools like ProVerif, the actual proof checking is done by running a computer on the models. Therefore the reviewer's main task is to determine whether the models checked are appropriate, based on both the design documentation and the requirements of the Federal Chancellery. Hence I strongly recommend that all future models make these connections even more explicit and direct.

With respect to the design document, the relationship between data manipulated by the model must be directly related to the data computed with by the design, and all computations

(transitions) in the model must be explicitly related to computations described in the design. Ideally this would be done by ensuring the following.

1. The names of all data items, such as keys, given in the design document (SysSpec) precisely correspond to similarly named data items in the ProVerif model.
2. All steps taken in the ProVerif model are reflected in the design and vice versa. Alternatively, any omissions and additions are explicitly commented on. This could be done by commenting each process step in the ProVerif model with where (e.g., page/line or position in message-sequence chart) it is described in the design.
3. When computations are simplified or changed, e.g., due to the use of abstractions in modeling, this should also be commented on. In this way, there would be much greater clarity about the relationship between the design and the model as a mathematical abstraction of it.
4. The properties and trust assumptions should be accompanied by an explanation of how they model the Federal Chancellery's requirements (which was the case here).
5. When multiple models are given, they should be as similar as possible, with the exception of the trust assumptions and the properties proven.

Documenting the relationship between the design and the model should ideally be included in the appropriate places of the model itself, rather than in a separate document. This would greatly simplify the task of checking the correspondences.

Following these recommendations would require substantially more work during the modeling phase. Much of the model would then be comments and care and effort would be needed to update the model and comments with each design change. This would shift much of the work from the auditor to the modeler, which is appropriate given that the modeler knows best what the relationships are. Moreover, it would ensure that the modeler takes full responsibility for ensuring model coherence as the design evolves.