Berner Fachhochschule (BFH), CH-2501 Biel, Switzerland

Examination of the Swiss Post Internet Voting System

Scope 1: Cryptographic Protocol

Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis

March 28, 2022

On behalf of the Federal Chancellery

Revision History

Revision	Date	Description
0.1	12.07.2021	Document initialization.
0.2	13.08.2021	Interim report submitted to Federal Chancellery.
0.3	28.11.2021	Draft submitted to Federal Chancellery.
1.0	28.03.2022	Final version submitted to Federal Chancellery. Some clarifications added after receiving feedback from the Federal Chancellery and from Swiss Post. Minor textual improvements.

Contents

Management Summary

In our assessment of the Swiss Post e-voting protocol, we observed that many improvements have been made compared to earlier versions. The protocol as such is still relatively complex and difficult to understand, but the current presentation in the available public documents has been improved substantially. There are also many issues that have been reported to the Swiss Post at earlier occasions, which have now been addressed on the current version. Generally, we can confirm that the development process of the cryptographic protocol points into the right direction. The assessment process itself benefits from the fact that the whole system is now owned and managed by Swiss Post, which greatly simplifies the communication between the experts and the product owner. We also support the decision of publishing all relevant documents at regular intervals.

We can also confirm that the process of involving an international group of e-voting experts at a much earlier stage has been very beneficial. Collectively, the findings listed in this report and the issues reported by the other experts represent a great source for further improvements. Implementing these improvements as early as possible will help to greatly reduce the chance of making further unexpected discoveries along with negative publicity, as it happened for example in the aftermath of the public intrusion test conducted in 2019. The decision to involve experts in assessing the protocol is therefore also an important measure for establishing and consolidating public trust.

Some of the findings listed in this report are issues that should be addressed before using the protocol in real elections. A particular concern is the vote privacy problem that results from the proposed way of processing the ballots from voters with restricted eligibility. The fact that only a minority of voters is affected does not diminish the problem, on the contrary, it accentuates the problem. The resulting privacy impact is particularly critical, because vote secrecy is broken as part of the final public result, i.e., without an adversary conducting an actual attack. We propose a better solution, in which the cause of this problem is completely eliminated.

Many of the other findings are in areas of the current specification document that are clearly not yet sufficiently concise. This includes a number of underspecified aspects of the cryptographic protocol, which offer too much room for interpretation and therefore may lead to the introduction of unexpected security problems later in the development process. Improving these areas of the document is straightforward in most cases, but we see it as a mandatory step that cannot be postponed.

As a general impression at the end of our assessment, it appears that the protocol and the available specification document have not yet reached the necessary maturity level that one would expect to see at this stage of the process. The most obvious example underlining this impression are the outdated references to the current OEV, which differs from the relevant draft OEV in some important points. This creates a number of inconsistencies, for example with respect to the roles and communication

abilities of the involved parties. Such problems are at the same time unnecessary and unacceptable, given the importance of the topic.

1 Introduction

This examination report lists the findings of our assessment of the Swiss Post e-voting protocol. We have been assigned with this task by the Federal Chancellery in June 2021 for a period of 6 months. While parts of this document have been given to the Federal Chancellery as an interim report on August 13, a draft of the full document has been released on November 26. The feedback that we received from both the Federal Chancellery and the Swiss Post allowed us to finalize the document. The content of this document has been worked out jointly by the listed authors from the Bern University of Sciences and independently of any other group of people. During our mission, we have been in loose contact with both the Federal Chancellery and the Swiss Post, mainly for obtaining clarifying information on certain topics.

1.1 Relevant Documents

To conduct our assessment, we received two documents from the Federal Chancellery and one document from the Swiss Post Ldt. The legal ordinance from the Federal Chancellery and its annex are accompanied by an explanatory report, which contains additional clarifying information. The Swiss Post protocol specification is largely self-explanatory, but it contains some links to further documents available on their website.[1] We are aware of the contents of these documents, but we will not refer to them in our report. The relevant documents for this report are therefore the following:

- [DraftOEV] *Federal Chancellery Ordinance on Electronic Voting*, Federal Chancellery FCh, Draft of April 28, 2021 (with Annex on Technical and Administrative Requirements for Electronic Voting).
- [ExpRep] *Partial Revision of the Ordinance on Political Rights and Total Revision of the Federal Chancellery Ordinance on Electronic Voting (Redesign of Trials) – Explanatory Report for Consultation*, Federal Chancellery FCh, April 28, 2021.
- [ProtSpec] *Protocol of the Swiss Post Voting System – Computational Proof of Complete Verifiability and Privacy*, Version 0.9.10, Swiss Post Ltd., June 25, 2021.

During the writing of this report, [DraftOEV] went through a public consultation process. According to a press release on December 10, 2021, the Federal Council has decided to finalize and publish the new ordinance in mid-2022. Given the large amount of responses to the consultation, the final documents are likely to contain some changes.

To emphasize its state as a non-final document, we refer to it as "draft OEV" (as opposed to "current OEV", which we will sometimes use to refer to the current ordinance from 2018). For understanding the requirements defined in [DraftOEV] and [ExpRep] as precisely as possible, we have mainly looked at the official document versions in German.

[1] See https://evoting-community.post.ch/en/community-programme

However, the terminology and citations used in this document are all taken from the available English translations.

1.2 Purpose and Scope of Examination

In a document called "*Audit Concept v1.3*", the Federal Chancellery describes the rules for preparing, conducting and reporting the examination. This document has been given to both the examiners and the examinees. It defines the general purpose of the examination as follows:

> *"In the context of the assessment of the Swiss Post system, the experts shall answer the following questions:*
>
> - *Are the system, its development and operation compliant with the legal requirements [...]?*
> - *Are the measures taken to mitigate risks effective?*
> - *Which improvements could be made for the sake of security, trust and acceptance?"*

The same document also defines the specific examination purpose for Scope 1:

> *"The protocol must fulfill the requirements listed in Chapter 2 of the annex of the draft OEV."*

The main goal of our assessment in Scope 1 is therefore to locate potential deviations between the cryptographic protocol as specified in [ProtSpec] and the requirements of the draft OEV and its annex. In case of encountered problems, our mission also includes making proposals for improvements.

In another document entitled "*Mapping VEleS Anhang*", the Federal Chancellery precisely defines the items from the annex of the draft OEV to be examined in Scope 1. In our systematic analysis in Section 3, we used the content of this document as a template.

2 Critical Findings

In this section, we provide an overview of the most critical findings of our assessment. In each case, we believe that the current protocol specification is not yet mature enough to achieve the necessary level of compliance with the legal requirements listed in the draft OEV. By improving the clearness and level of details of the protocol specification, some of the discussed issues could probably be addressed or eliminated rather easily. Other points, however, may possibly require a redesign or extension of certain protocol aspects. We understand locating such critical points in the protocol as the main goal of our assessment, but we will also present some ideas for possible workarounds and provide a few recommendations for improvements.

Most of the discussed issues have an impact on multiple items listed in the draft OEV. Whenever useful, we include corresponding cross-references to demonstrate the impact of the finding. More specific findings will be listed in Section 3, which follows the given structure of the draft OEV and includes all relevant points for Scope 1.

2.1 Missing Update to Draft OEV

The current version of the protocol specification has obviously not yet been updated to the draft of the new legal ordinance. Since all references to the ordinance and the corresponding technical annex explicitly refer to the current version from July 2018 (cited as [19] and [20], respectively), it is clear that numerous synchronization conflicts exist throughout the document. We understand that most aspects of the protocol have been designed long before the draft of the new ordinance has been released in April 2021, but in order to conduct the assessment, we would have expected to receive in time a synchronized version of the specification document that is aligned with [DraftOEV].

Clearly, the lack of such an update unnecessarily complicates the evaluation of the protocol, because it forces readers of the protocol specification to speculate about how to bring certain discrepancies between the documents into alignment. To conduct our assessment without delay, we performed such alignments on a best-effort basis based on our pre-knowledge about the protocol and its most recent adjustments. This said, we are concerned of having spent efforts on unfinished work, which as such should not have been accepted into the evaluation process. The fact that we received a document versioned as 0.9.10, which usually refers to an incomplete and unstable pre-version of the initial release, underlines our concern.

From the perspective of cryptographic protocol design, a notable change in the draft OEV is the explicit introduction of an additional fully trusted component for setting up an election. The so-called *set-up component* can be used to prepare the polling cards, which includes generating the voters' secret authentication and verification codes [ExpRep, Sect. 5.2.2, No. 2.1]:

> *"Set-up component: [...] The canton prepares data for the ballot using the set-up component. This includes data whose randomness and confidentiality are crucial to achieving the requirements for the cryptographic protocol [...], such as the voters' verification reference."*

The second fully trusted component in the draft OEV is the *print component*, which is operated by the *printing office*. According to [DraftOEV, Annex 2.1] and [ExpRep, Sect. 5.2.2, No. 2.1], the task of the print component is restricted to printing, packaging, and mailing the polling cards to the voters:

> *"Print component: It prints the verification reference for the voters. This abstract term includes packaging and mailing to voters. [...]"*

Furthermore, according to [DraftOEV, Annex 2.2], no outgoing communication is allowed other than from the print component to the voters. The illustration in Figure 1 shows the permitted communication model. It is taken from [ExpRep, Sect. 5.2.2, No. 2.2].

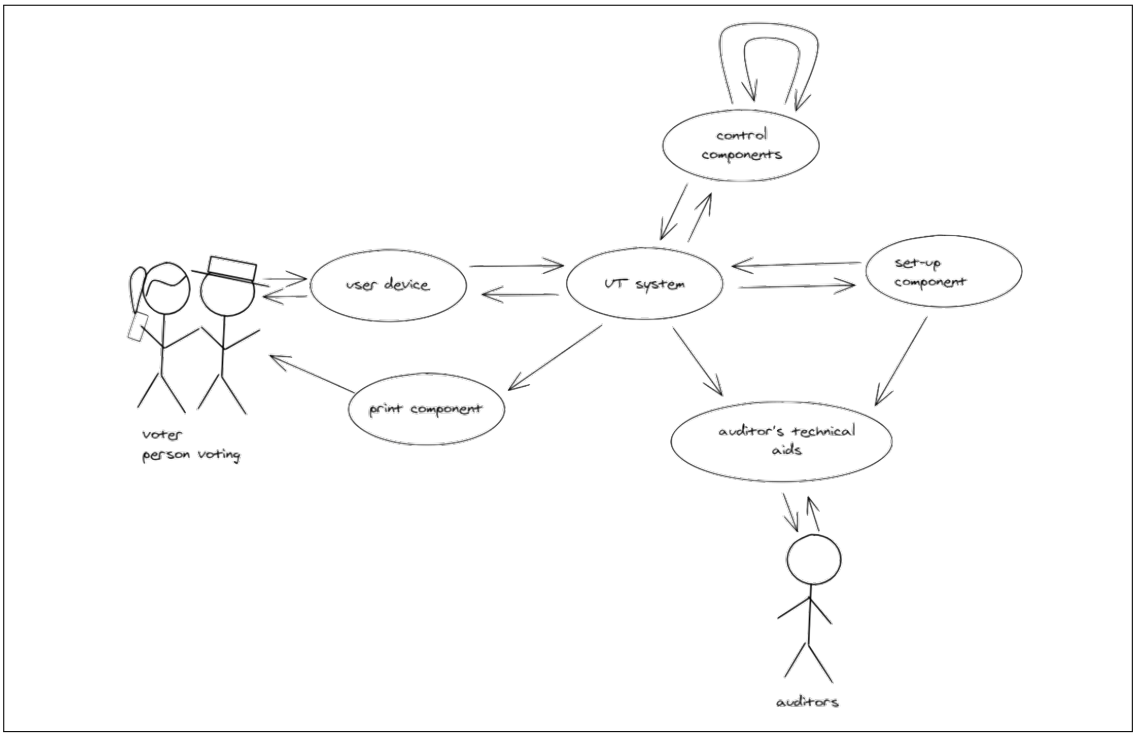

Figure 1: The communication model as defined in [DraftOEV, Art. 2.2] and [ExpRep, Sect. 5.2.2].

In the protocol specification, the tasks of preparing and printing the polling cards are conducted by the *print office* alone. By interpreting the terms "print office" from [Prot-

Spec] and "printing office" from [DraftOEV] as synonyms, we get the most obvious discrepancy between the two documents, because the print office, according to its role specified in [ExpRep, Sect. 5.2.2], is not allowed to perform most of its assigned tasks, including all tasks implemented by the following algorithms [ProtSpec, Sect. 11.1]:

- GenVerCardSetKeys
- GenEncryptionKeysPO
- GenVerDat
- CombineEncLongCodeShares
- GenCMTable
- GenCredDat
- SetupTallyPO

In addition to violating the allowed scope of concern, the print office uses multiple communication channels that are not permitted in [DraftOEV]:

- Print Office $\longrightarrow$ Voting Server
- Print Office $\longrightarrow$ Auditors
- Print Office $\longrightarrow$ Electoral Board

To the best of our understanding, this design of the print office is due to the current OEV, which is less restrictive with respect to the printing office's responsibilities and communication abilities. This point has been extensively discussed as a potential weakness of the proposed protocol, for example during the 2020 expert dialogue on Internet voting in Switzerland, which was initiated by the Federal Chancellery. We assume that corresponding changes in the draft OEV result from these discussions.

To achieve compliance with the draft OEV, the print office as defined in [ProtSpec] simply needs to be decomposed into a set-up and a print component. By attributing all above-mentioned algorithms and communications to the set-up component and restricting the print component's responsibility to the printing process, the above-mentioned document synchronization problem could be solved rather easily.

However, we want to stress our concern of having a fully trusted set-up component, which generates all the secret codes necessary for submitting and verifying votes during the election. By attacking the set-up component, an adversary could use these codes for breaking both the integrity and privacy of the votes. To protect the set-up component from such attacks, we recommend simplifying the design of the set-up component to the maximum possible extent, for example by generating the secret codes in a fully distributed manner by the control components or by further restricting the available communication channels. In the simplest possible case, the set-up component receives the control components' shares of the secret codes, assembles them using a deterministic procedure, and forwards the obtained polling card data to the print component.

2.2 Role of Auditors

Regarding the auditor's role in the protocol, we encountered several critical problems. Generally, the idea of conducting verifiable elections implies the existence of at least one person performing the verification. In the least restricted setting of a public verification, anyone equipped with corresponding tools and sufficient technical knowledge can verify the election based on the published election data [1]. In the literature on verifiable elections, the person conducting the verification is usually called *verifier*. This term incorporates the person's technical toolbox, which is needed to perform the verification in an independent process based on the available election data. In the simplest case, the verifier outputs a binary value $v \in \{ok, \neg ok\}$ stating whether the verification has been successful or not (failure cases may require more sophisticated outputs with details about the cause of the failure). The English draft OEV translation uses the term *auditor* instead of verifier, but the assigned role and tasks are defined analogously [DraftOEV, Art. 2, Para. 1h]:

> *"Auditor means a person who checks on behalf of the canton that the ballot is correctly conducted."*

An even more detailed definition of the auditor's role in the protocol is given in the explanatory report. It distinguishes the auditor as a person from the auditor's technical aid [ExpRep, Sect. 5.2.2, No. 2.1]:[2]

> *"Auditors: After tallying, the auditors receive a proof from the UT system [...] which confirms that the results have been tallied correctly. They conduct the check at least once with a technical aid. During the setup phase, they can also use their technical aid to perform checks on behalf of the setup component."*

Note that the last sentence of this definition is an important change from the current OEV to the draft OEV, because it enhances the auditor's possible responsibility of checking the election result after tallying to performing additional checks during the setup phase. While such an enhanced role deviates from the common understanding of the verifier's role in the e-voting literature, it is not in conflict with the general goal of detecting all possible manipulations of an election.

By explicitly allowing the auditors to perform their checks "during the setup phase" and "after tallying", we understand the above definition as a prohibition for the auditors to act in other phases of the election process, for example during the voting or tallying

[2] By calling the auditor's technical aid *verifier*, the protocol specification also makes a distinction between the person and the machine (software) performing the verification. However, the definition of the verifier in [ProtSpec, Sect. 2.1.1, Table 1] seems to be needless, because no such component appears in the protocol description of [ProtSpec, Sect. 11].

phase.[3] Under this premise, the current protocol specification deviates from the draft OEV in [ProtSpec, Sect. 11.2, Fig. 23] and in [ProtSpec, Sect. 11.3, Fig. 24], where the auditors are involved during the voting and tally phase, respectively. Here, a redesign of the auditors' role in the protocol seems necessary.[4]

Another problem of the auditors in the protocol specification is the additional task assigned in [ProtSpec, Sect. 11.3, Fig. 24], which consists in receiving $\mathbf{c}_{\mathsf{Dec},m'}$ from CCM $m'-1$ and forwarding it to CCM m' only in case all checks have succeeded. By strictly interpreting the auditor's role as a party entitled to perform checks and announce corresponding results, this additional task is clearly not compatible with the above definitions from the draft OEV and the explanatory report. Furthermore, neither the auditors nor their technical aids are allowed to directly communicate with any other component of the whole system (see Figure 1). We understand this requirement in the sense that auditors will report the result of the verification process to someone from outside the communication model, for example the cantonal election authorities, who will intervene in case of irregularities. In any case, the communications from the auditors to the print office in [ProtSpec, Sect. Fig. 19], from the auditors to the CMMs in [ProtSpec, Sect. Fig. 23], and from the auditors to the electoral board and the last CCM in [ProtSpec, Sect. Fig. 24] are clearly in conflict with the draft OEV, even if the transmitted information consist of a single bit $v \in \{ok, \neg ok\}$ only.

The last problem with the auditors in the protocol results from the fact that they are defined as a group of people, of which at least one is trustworthy [ProtSpec, Sect. 2.1.1]. But what is completely unspecified is the decision making process between the members of this group, which is critical in cases where they disagree. If a majority rule is applied in such cases, then a single trustworthy auditor could always be overruled by two or more untrustworthy auditors, i.e., the assumption of at least one trustworthy auditor would then not prevent an attack by the untrustworthy auditors. The best way to solve this problem would probably be to trigger an investigation in cases where at least one auditor disagrees. The problem in the current protocol specification is the absence of an appropriate discussion of this problem.

[3] Clearly, performing some preliminary computations based on already available data is always permitted without any restrictions and at the earliest possible stage of the process, as long as it does not interfere with the current protocol execution.

[4] We are aware of the remark on vote secrecy given in [ExpRep, Sect. 2.9.3], which seems to allow using the verifier's trustworthy technical aid before tallying for the purpose of protecting vote secrecy: *"By this time, however, voting secrecy would already have been breached. This must be prevented by having trustworthy components ensure that no marked votes are processed before tallying. In view of this objective, a technical aid used by the auditors may also be considered trustworthy."*

In the light of this remark, the design decision of the Swiss Post e-voting protocol seems to be appropriate. In our opinion, however, measures to enable universal verification should not be mixed up with measures to protect vote privacy, even if they are explicitly permitted by the legal requirements. From a protocol design perspective, we see it as a violation of respective responsibilities, which complicates the protocol implementation for no obvious reasons. Therefore, we recommend that future versions of both [ExpRep, Sect. 2.9.3] and the Swiss Post e-voting protocol will no longer include this possibility.

2.3 Proving Vote Abstention

Ballot stuffing—especially the case of adding fraudulent ballots for abstaining voters—is a major concerns in both non-electronic and electronic voting systems. While individual verifiability in e-voting is usually understood as the ability of voters to check that their ballot has been cast and recorded as intended, the draft OEV goes even a step further by allowing abstaining voters to perform a similar check [DraftOEV, Art. 5, Para. 2c]:

> *"A voter who has not cast his or her vote electronically can request proof after the electronic voting system is closed and within the statutory appeal deadlines that the trustworthy part of the system has not registered any vote cast using their client-side authentication credential."*

A second definition is given in [DraftOEV, Annex 2.5], which defines the requirement for the cryptographic protocol with respect to individual verifiability:

> *"The voter is given a proof [...] that the attacker [...] has not maliciously cast a vote on the voter's behalf which has subsequently been registered as a vote cast in conformity with the system and counted."*

In its full length, the last quote includes a reference to [DraftOEV, Art. 6], which defines the meaning of the term "proof" in this particular context as follows:

> *"The soundness of the proof under Article 5 is based exclusively on the trustworthiness: a) of the trustworthy part of the system for proof under Article 5 paragraphs 2 and 3; b) of the procedure for generating and printing the voting papers for proof under Article 5 paragraph 2;"*

Finally, [DraftOEV, Annex 2.11.1] defines an upper limit for an attacker's chance of falsifying such a proof:

> *"The probability of the attacker being able to falsify a proof under Number 2.5 if he changes a partial vote, suppresses a partial vote or casts a vote in someone else's name must not exceed 0.1%."*

To the best of our understanding, all these definitions and requirements apply to both types of proof, the ones issued to the participating voters and the ones issued to the abstaining voters. Under this premise, we would expect from the protocol to offer similar implementations for both types of proof, for example by extending each polling card with an additional *vote abstention code*.[5] However, this feature is entirely missing in the current protocol specification.

We propose to implement vote abstention codes in a procedure similar to the existing vote cast return codes, ideally by generating them by the group of control components

[5] The possibility of using vote abstention codes for offering individual verifiability to non-participating voters has been explored in [2, Sect. 2.1].

in a distributed manner. Note that by using the same character set and code length, the two types of codes become indistinguishable. In such a scenario, displaying the corresponding code (vote cast return code or vote abstention code) to a voter in the aftermath of an election does not leak the information about the voter's participation to the voting client. Therefore, for optimizing the usability of this check, one could even think of publishing the list of codes of all eligible voters together with the election result, for example on the canton's official election website.[6]

2.4 Understudied Protocol Aspects

In the requirements for the definition and description of the cryptographic protocol, the draft OEV is much stricter than its predecessor. Specifically, [DraftOEV, Annex 2.13.2] requires the protocol specification to narrow down the possible implementation choices to the point where violating the given cryptographic protocol requirements is no longer possible:

> *"Instructions must not be underspecified. Individual instructions must restrict the options for implementation to such a degree that any form of implementation that the instructions allow is also compliant with meeting the cryptographic protocol requirements."*

We understand this requirement in the sense that all cryptographically relevant aspects of the protocol must be specified in great detail. By describing a large number of algorithms precisely in an almost pseudocode-like manner, [ProtSpec, Sect. 12] fulfills this requirement for most parts of the protocol. Clearly, real pseudocode would narrow down the implementation options even further, but the given level of details seems sufficient to us in most cases.

At some places, however, the protocol specification leaves too much room for interpretation. Some of the encountered problems have the potential of compromising the election result or vote secrecy, for example if corresponding units—by accident or on purpose—are not implemented correctly. The most critical problems of that kind are listed in the following subsections.

[6] We are aware of the following remark in [ExpRep, Sect. 2.5] about how to efficiently implement the required proof in a practical system: *"For reasons of efficiency, it is sufficient for the competent cantonal office to confirm to the voter that no vote has been cast on their behalf."* Thinking of implementing this process at the cantonal offices, for example using a hotline with trained poll workers answering the phone calls, we are concerned that both the initial and recurring costs will be considerably higher than in our proposed solution with indistinguishable finalization and abstention codes. These codes can be printed and published for all voters at low costs right after the election in an automated process and without any impact on security. In contrast to that, implementing a hotline requires hiring and training a sufficient amount of qualified people, which can even handle worst-case scenarios such as a denial-of-service attack against the hotline. Therefore, we believe that solutions like these provide the contrary of efficiency compared to our solution. They also come with extra trust assumptions about the reliability and robustness of the implemented processes at the cantons. Our recommendation for introducing vote abstention codes aims at achieving the best possible security for the lowest possible cost.

2.4.1 Primes Mapping Table

The generation of the *primes mapping table* pTable from [ProtSpec, Sect. 10.3] is completely unspecified. Since the table is an input for the tallying procedure and therefore is critical for the correctness of the election result, it is important to know exactly how, when, and by whom the table is generated. At the moment, pTable appears for the first time in the algorithm GenVerDat, which is executed by the print office during the setup phase.[7] It is then added to the print office's $\mathsf{Logs}_{\mathsf{PO}}$, which later is handed over to the auditors. It also appears as an input to the algorithms CreateVote (executed by the voting client) and DecodePlaintexts (executed by the last CCM). Unfortunately, the protocol does not specify how pTable reaches the print office, the voting client, and the last CCM.

Clearly, this problem could be solved easily by letting each party run the same deterministic algorithm for obtaining pTable, but no such algorithm is included in the protocol specification. Another solution would be to let a single party generate pTable during the setup phase and then transmit it to every other depending party. In this solution, each party receiving pTable must check that the mapping is unique and contains only primes from the underlying sub-group, because otherwise an incorrect pTable may ultimately lead to incorrect election results. Unfortunately, no such checks are specified in the protocol description. Furthermore, if an adversary manages to infiltrate different prime mapping tables at different steps during a protocol execution, the outcome of the election could be completely inverted. Note that similar problems may result from programming errors, inconsistent code versions, or faulty operation. Generally, not specifying the mapping between voting options and prime number representations prevents the system from being universally verifiable.

2.4.2 Electoral Board Key Pair

In the SetupTally protocol in [ProtSpec, Sect. 11.1], the print office generates the key pair $(\mathtt{EB}_{\mathtt{pk}}, \mathtt{EB}_{\mathtt{sk}})$ and sends it to the electoral board.[8] To protect the private key $\mathtt{EB}_{\mathtt{sk}}$ during its transmission over an insecure channel, it is symmetrically encrypted using the $\mathsf{ENC}_{\mathsf{s}}$ algorithm.[9] The problem here is that important information about exchanging or establishing the symmetric key between the print office and the electoral board is completely missing.

According to the OEV communication model (see Figure 1), no trustworthy channel exists between the print office and the electoral board, i.e., the symmetric key cannot

[7] We think that pTable should be added to the input parameters of GenVerDat, otherwise it is inconsistent with CreateVote and DecodePlaintexts.

[8] The idea of generating the electoral board's key pair by the print office is questionable from the perspective of good cryptographic design.

[9] Calling $\mathsf{ENC}_{\mathsf{s}}(\mathtt{EB}_{\mathtt{sk}})$ with a single parameter $\mathtt{EB}_{\mathtt{sk}}$ is clearly a mistake in the description of the SetupTally protocol, because according to its definition in [ProtSpec, Sect. 4], $\mathsf{ENC}_{\mathsf{s}}$ should have two parameters (the symmetric key is missing).

simply be generated by one party and sent to the other party. Alternatively, one could think of generating the symmetric key jointly by executing a key establishment protocol such as Diffie-Hellman, but this is not secure against an active man-in-the-middle. Our conclusion here is that without assuming the existence of a PKI that involves either the print office or the electoral board (or both), this problem cannot be solved properly.

2.4.3 Ballot Box

The ballot box bb is not sufficiently well defined. It first appears in [ProtSpec, Sect. 11.2] as input to the algorithms ExtractCRC (protocol SendVote) and ExtractVCC (protocol ConfirmVote). Both algorithms are executed by the voting server, and in both cases, executing the algorithm changes the bb's internal state as a side effect. Therefore, bb is mainly a data structure created and maintained by the voting server during the voting phase. Later in the protocol, bb is transmitted to the auditors as input for their algorithms VerifyVotingPhase and VerifyOnlineTally. Finally, the voting server uses bb as input for the algorithm Cleansing at the beginning of the tallying phase.

By looking at the algorithms called by the voting server during the voting phase, the ballot box can be interpreted as a data structure consisting of the two lists $\mathsf{L}_{\mathsf{sentVotes}}$ and $\mathsf{L}_{\mathsf{confirmedVotes}}$ consisting of pairs $(\mathtt{vcd}_{\mathtt{id}}, \mathtt{extCC})$ and triples $(\mathtt{b}_{\mathtt{id}}, \mathtt{attempts}_{\mathtt{id}}, \mathtt{extVCC})$, respectively. Later in the Cleansing algorithm, a new ballot box $\mathsf{bb}_{\mathsf{clean}}$ is created, which is a simple list of all remaining ElGamal encryptions.[10] After sending this list to all CCMs, the first CCM uses it as input to the algorithm MixDecOnline. This algorithm first performs a test $\mathsf{bb} \notin \mathsf{L}_{\mathsf{bb,j}}$ and later an update $\mathsf{L}_{\mathsf{bb},j} \leftarrow \mathsf{L}_{\mathsf{bb},j} \cup \mathsf{bb}$. Here, several points are either underspecified or wrong:

- It is not clear whether bb refers to the original ballot box from the voting server or to $\mathsf{bb}_{\mathsf{clean}}$. If it refers to the original ballot box, which is unknown to the CCMs at this point of the protocol, it is unclear how and when it is transmitted to the CCMs. If it refers to $\mathsf{bb}_{\mathsf{clean}}$, then the algorithm's description is wrong. In both cases, it should be added to the algorithm's parameter list.
- $\mathsf{L}_{\mathsf{bb},j}$ is undefined. According to the performed check and update operation, which includes set operations $\in$ and $\cup$, it seems that $\mathsf{L}_{\mathsf{bb},j}$ refers to a set of bb objects. In that case, updating this set should be described as $\mathsf{L}_{\mathsf{bb},j} \leftarrow \mathsf{L}_{\mathsf{bb},j} \cup \{\mathsf{bb}\}$. Alternatively, by interpreting bb as a set rather than a list (which is formally incorrect), $\mathsf{L}_{\mathsf{bb},j}$ would become the set of all elements from all bb's.

In either of the above cases, the purpose of $\mathsf{L}_{\mathsf{bb},j}$ and the performed test remains unclear. To the best of our understanding, its purpose is to check that every CCM shuffles only one list of encrypted votes per election event. Such a check seems to be important to prevent an attack on vote secrecy by an auditor colluding with the offline CCM and the untrusted voting server. However, this check does not fulfill its purpose, as it only

[10] Calling this list of encryptions "ballot box" is clearly not an optimal choice, because then the same term is used for quite different things.

guarantees that not exactly the same list of encrypted votes is shuffled and decrypted twice, i.e., cases of identical sub-lists remain undetected. Here, we expect more detailed explanations from [ProtSpec] and an improved description of this subject.

2.4.4 Logs

Similar to the ballot box, also the logs $\mathsf{Logs}_{\mathsf{PO}}$, $\mathsf{Logs}_{\mathsf{CCR},j}$, and $\mathsf{Logs}_{\mathsf{CCM},j}$ are not sufficiently well defined. To the best of our understanding, they represent the parties' state of knowledge at different stages of a protocol. During the course of the protocol execution, all logs are transmitted to the auditors, which require them as inputs for their verification algorithms. In most cases, the contents of these logs can be guessed from looking at all algorithms that change the logs' internal states.

If strictly implemented as specified in these algorithms, many checks involving the logs do not fulfill their intended purpose. For example in $\mathsf{Logs}_{\mathsf{CCR},j}$, the list $\mathsf{L}_{\mathsf{decPCC},j}$ as defined in algorithm DecryptPCC is a set of $(\mathtt{vcd}_{\mathtt{id}}, \mathbf{pCC}_{\mathtt{id}})$ tuples. But this implies that checking $\mathtt{b}_{\mathtt{id}} \in \mathsf{L}_{\mathsf{decPCC},j}$ in CreateLCCShare will never succeed, because the involved elements are incompatible. Another example is $\mathsf{L}_{\mathsf{sentVotes},j}$ from $\mathsf{Logs}_{\mathsf{CCR},j}$, which according to CreateLCCShare is a list of $(\mathtt{b}_{\mathtt{id}}, \mathbf{pCC}_{\mathtt{id}}, \mathbf{lCC}_{j,\mathtt{id}}, \pi_{\mathsf{exp},j})$ tuples. But this structure is clearly incompatible with checking $\mathtt{vcd}_{\mathtt{id}} \notin \mathsf{L}_{\mathsf{sentVotes},j}$ in CreateLVCCShare, which as a consequence will always return *true*.

Given the importance of the logs in the design of the protocol, problems of the above types are unacceptable in the protocol specification. In each of the three cases, we would recommend to explain the purpose and structure of the logs much more accurately and to ensure that their applications are always described correctly.

2.4.5 Keystore and Start Voting Key

The protocol as presented in [ProtSpec] does not specify when and how the voting client receives the keystore $\mathtt{VCks}_{\mathtt{id}}$. In the protocol SetupVoting, the list **VCks** of all keystores is generated by the print office and transmitted to the voting server. According to algorithm GenCredDat, the symmetric keys $\mathtt{KSkey}_{\mathtt{id}}$ used to encrypt the secret keys $\mathtt{k}_{\mathtt{id}}$ are derived deterministically from the start voting keys $\mathtt{SVK}_{\mathtt{id}}$ using a password-based key derivation function, and the resulting keystores $\mathtt{VCks}_{\mathtt{id}}$ are collected in **VCks**. Later in the protocol SendVote, the voting client uses $\mathtt{VCks}_{\mathtt{id}}$ as input to the algorithm GetKey, which performs the corresponding symmetric decryption. The fact that the transmission of $\mathtt{VCks}_{\mathtt{id}}$ from the voting server to the voting client is not specified in the protocol description is possibly an unintended mistake, but it is an important detail that needs to be specified.

A different kind of problem is related to the start voting key $\mathtt{SVK}_{\mathtt{id}}$, which is defined as a 20-character Base32 password. These passwords are are used in a PBKDF2-based key derivation function with a fixed amount of $R = 32$'000 iterations [ProtSpec, Sect. 6].

Unfortunately, both the password length $|\mathtt{SVK}_{\mathtt{id}}|$ and the number R of iterations are independent on the security level. This implies a fixed amount of approximately 115 bits security.[11] Note that this is less than the 128 bits security of the *extended* security level as defined in [ProtSpec, Sect. 20.1]. Here, the lack of proper parametrization thus undermines the protocol's own security concept.

2.4.6 Authentication and Context Separation

For a cryptographic protocol to provide its security guarantees, it is important to add strong authentication to the available communication channels. In the given context, where elections with different electorates are held simultaneously (see previous subsection), there are two aspects to consider. First, parties need to know to *whom* they are talking to. This is the classical authentication problem, which can be solved in different ways. The goal is to guarantee that no adversary can create or modify messages without being detected by the recipient. Second, parties need to know *what* they are talking about. This is important for understanding the meaning and context of a received message. For example, receiving the message *ok* from the auditors has completely different meanings in the setup, voting, and tally phases, or even across different protocol runs. This problem known as *context separation* needs to be addressed properly in the design of secure protocols to avoid many sorts of replay or cross-protocol attacks.

Strong authentication in the above sense can be implemented rather easily by signing all protocol messages together with corresponding *context strings*. To prevent context-separation problems in all possible cases, we recommend making context strings unique over all protocol messages and all protocol executions, for example by concatenating unique message IDs with unique protocol execution IDs. If m is a message to transmit from one party to another and ctx a unique context string, then $\sigma = \mathsf{sign}_{sk}(m||ctx)$ is the signature that needs to be sent to the recipient along with m. Upon receiving (m, σ), the recipient conducts the processing of m only after successfully checking the validity of σ. Implementing strong authentication in this way for all involved system components is a sufficient measure to overcome this type of problems. It ensures that at every protocol step the right messages are processed and only for the intended purpose.

Authentication is not sufficiently well described in the protocol specification. The overview given in [ProtSpec, Sect. Table 6] lists the channels, for which authentication based on digital signatures needs to be implemented. First, we believe that this list is incomplete, because other important channels should be protected as well:

- Print Office $\longrightarrow$ Auditors (see Fig. 19)
- Print Office $\longrightarrow$ Electoral Board (see Fig. 20)
- CCRs $\longrightarrow$ CCRs (via Voting Server, see Fig. 21)

[11] By counting 32'000 iterations as $\log(32000)_2 \approx 15$ additional security bits, we obtain $20*5+15 = 115$ bits security.

- CCRs ⟶ Voting Client (see Figs. 21, 22)
- CCRs ⟶ Auditors (see Figs. 23)
- CCMs ⟶ CCMs (via Voting Server, see Fig. 24)[12]

Generally, we recommend adding message signatures systematically to all outgoing messages of at least the Print Office, the CCRs, and the CCMs.[13]

Second, we think that important information about solving the context separation problem is missing. In [ProtSpec, Sect. 13.1.1], an *election event ID* $\texttt{ee}_{\texttt{id}}$ is introduced to "ensure that each run is unique", but given that multiple protocol runs are conducted simultaneously for different subsets of the electorate of a given election event (see Subsection 2.5), this is clearly not sufficient. As mentioned above, we recommend using context strings consisting of unique message type and protocol execution IDs. Given its importance for the protocol's security properties, this aspect definitely needs to obtain more attention in the protocol specification.[14]

Finally, we recommend including the details of the digital signature scheme into the protocol description. From the *System Specification* (Scope 2), we know that the RSA-PSS signature scheme with 2048-bits key length is used in the actual implementation. One problem with this particular choice is the fixed key length, which is only sufficient for the *default security level* from [ProtSpec, Sect. 20.1], but not for the *extended security level*. The fact that the security of the cryptographic protocol is properly parameterized implies that all cryptographically relevant components must be aligned accordingly. This means that selecting the signature key length cannot be delegated to the system developers, who may be unaware of the cryptographic impact for the whole system.

2.4.7 Election Use Cases

The presentation of the protocol in [ProtSpec] does not include precise descriptions of the manifold election types and election use cases from the given Swiss context. Many subtleties need to be taken into account to cover them all properly, for example the fact that federal, cantonal, and communal elections and referendums are often held simultaneously and that some voters may not be eligible to all of them. A good overview of the Swiss election use cases is given in [2, Sect. 2.2], which demonstrates that all election types (except those permitting write-ins) can be reduced to a general *election event* with t simultaneous k_j-out-of-n_j elections. Therefore, corresponding vectors $\mathbf{k} =$

[12] In Fig. 24, it is unclear how the result of a single mixing step is forwarded to the next CCM. We assume that this communication goes over the voting server, similar to the CCRs in Fig. 21.

[13] In Tables 17 and 18, the *System Specification* (Scope 2) provides a more detailed overview of the message signatures with an enhanced list of authenticated channels. In comparison with the list given above, the overview is still incomplete.

[14] Some information about solving the context separation problem is given in the *System Specification* (Scope 2). Statements like "The table omits the context information such as the election event ID that allows the participants to prevent replay attacks" show that the awareness of the problem is given, but also that the amount of given information is insufficient for properly describing this important issue.

$(k_1, \ldots, k_t)$ and $\mathbf{n} = (n_1, \ldots, n_t)$ are the main parameters for describing a general election event.

The electorate of such an election event is specified by the number of voters N and a Boolean matrix $\mathbf{E} = (e_{ij})_{N \times t}$ of values $e_{ij} \in \{0, 1\}$ describing the eligibility of voter $i \in [1, N]$ in election $j \in [1, t]$. In combination, $\mathbf{k}$, $\mathbf{n}$, and $\mathbf{E}$ define for each voter the number of allowed selections in each election and the number of available voting options. Clearly, for ensuring the correctness of an election outcome, it is important for all system components to know exactly these parameters at all times.[15]

In the given protocol description in [ProtSpec], election events are specified by the following parameters:

- ψ = number of allowed selections,
- n = number of voting options,
- N = number of voters,
- **correctnessID** = $(\text{correctnessID}_1, \ldots, \text{correctnessID}_\psi)$, where each correctnessID_i is a reference to one of the t simultaneous elections.

The last point of the above list is described in [ProtSpec, Sect. 10.4].[16] Compared to the above general model, we have the following obvious relationships:

- $\psi = \sum_{j=1}^{t} k_j$,
- $n = \sum_{j=1}^{t} n_j$,
- $t = |\{\text{correctnessID}_1, \ldots, \text{correctnessID}_\psi\}|$.

Furthermore, it is possible to derive $\mathbf{k} = (k_1, \ldots, k_t)$ from **correctnessID** by counting the number of appearances of each of the t distinct values. However, it seems that neither $\mathbf{n}$ nor $\mathbf{E}$ is included in the protocol's election model. Without knowing $\mathbf{n}$, it is impossible to assign the voting options unambiguously to the elections, and without knowing $\mathbf{E}$, it is impossible for the control components and the auditors to distinguish

[15] There are several possibilities that voters have differing voting right. One example are Swiss citizens living abroad. They are allowed to participate in federal referendums and elections, but many cantons restrict their voting rights for cantonal or municipal issues. In some places, the same rule applies for a certain period of time to citizen moving from one municipality or canton to another. Another example of voters with restricted eligibility are foreigners living in Switzerland. Some cantons offer them the right to vote for cantonal or municipal issues, but they are not allowed to participate in federal referendums and elections.

[16] We think that the overall clarity of this subsection should be improved. For example, by stating "[...], we introduce a vector **correctnessID**, containing an ID for each question and election", we conclude that this vector contains n elements, but later in the example, the length is clearly $\psi = 5$. In the algorithms GenVerDat, GenCMTable, CreateLCCShare, and ExtractCRC, where **correctnessID** is used, the length is twice indicated as n and twice as ψ. Therefore, it seems that something is either wrong or missing. By looking at the *System Specification* (Scope 2), we learned that there are actually two **correctnessID** vectors with slightly different names and different semantics, one of length n and one of length ψ. This is therefore an example, where the protocol and the system specifications are clearly not aligned properly.

voters with different voting rights in a combined election event. These important points should be further clarified in the protocol specification.

2.4.8 Write-Ins

Write-in candidates are completely omitted in the specification. In [ProtSpec, Sect. 13.3], it is claimed that the allowance of write-in candidates does not impact the security of the protocol and hence can be omitted in the description and the security proofs. We do not agree on this, because even the smallest modification in a cryptographic protocol may have a high impact on the protocol's security properties. For example, reusing the vote encryption key and randomization for encrypting the write-in candidates (e.g., for performance reasons) would jeopardize vote secrecy directly.

Furthermore, with respect to the definition of a *vote cast in conformity with the system* [DraftOEV, Art. 2.1p], allowing write-in candidates in a ballot may have an impact on accepting votes as valid or not, for example in case of a malicious voter submitting a vote for regular candidates in combination with non-empty fields for write-in candidates. To avoid the complexity of this difficult topic and to eliminate the potential risk for the protocol's security properties, we recommend removing write-ins (which appear only in some exceptional cases) entirely from the protocol specification.

2.5 Vote Privacy of Voters With Restricted Eligibility

From the *System Specification* (Scope 2), we know that voters with different voting rights are handled by defining corresponding subsets of the electorate (called *verification card sets*). Different instances of the protocol are then executed simultaneously for each of these subsets. We think that this design should be better justified, because it implies a severe privacy problem in cases where only a few members of the electorate have divergent voting rights (see Footnote 15 for examples that are likely to happen especially in small municipalities). In such a case, the election result computed separately for the corresponding subset of voters may reveal sufficient information for completely breaking the secrecy of the submitted votes. This clearly violates to the requirements on *voting secrecy* in [DraftOEV, Art. 7] and [DraftOEV, Annex 2.7.1], even if only a small portion of the electorate is affected:

> *"It must be ensured that the attacker is unable to breach voting secrecy or establish premature results unless he can control the voters or their user devices."*

From a more general perspective, introducing subsets of the electorate creates anonymity sets that are smaller than strictly necessary. In traditional Swiss elections, anonymity sets are created for each municipality by publishing their election results individually. With the introduction of e-voting, channel-specific election results must be published

for each municipality. This already reduces the size of the anonymity sets, but it corresponds to a requirement from the given legal framework [DraftOEV]. We believe that diminishing the size of the anonymity sets even further should be avoided, both from a legal and a vote secrecy perspective.

To fully circumvent this type of problem, we recommend a different approach in which the electorate of a municipality is kept together under all circumstances. The idea is to include in each ballot up to three different vote encryptions, one with the votes for federal issues, one with the votes for cantonal issues, and one with the votes for communal issues (in the current protocol, all votes are contained in a single encryption).[17] Depending on the voter's specific voting rights, control components and auditors can then checks if a ballot contains the right amount of encryptions. For this, an additional election parameter similar to **E** from Subsection 2.4.7 must be added to the election model and distributed to all system components. In the cleansing and mixing phase, these encryptions are extracted from the ballots and then processed independently through separate mix-nets. After the mixing, voters with restricted voting rights are no longer distinguishable from unrestricted voters. Clearly, this approach perfectly preserves the given anonymity sets and therefore no longer unnecessarily impacts vote secrecy as in the proposed solution. Note that the practice in certain cantons with different envelopes for federal, cantonal, and communal votes is an equivalent solution for paper votes.

2.6 Legitimacy of Proof

The proof as presented in [ProtSpec, Sect. 14 to 19] has a number of shortcomings and ambiguities. Overall, the proof is not fully convincing and hence it remains unclear, and in fact questionable, whether the protocol indeed has the claimed security properties. Besides not following the usual terms and definitions from the literature, the proof is too strongly abstracted from the real protocol. Here are a few of our observations:

- The proof is difficult to read and understand. There are plenty of undefined variables, deficiencies in syntax and unclear explanations. Sometimes, readers can guess from the context what is meant, but for a thorough examination, the room for interpretations should be as little as possible.
- For individual verifiability, the two properties *sent-as-indented* and *recorded-as-confirmed* are proven. However, for a complete verifiability chain, the property *confirmed-as-sent* seems to be missing. It is also surprising and not explained why the proof deviates from the common definition in literature, which defines individual verifiability as the chaining of two properties *sent-as-intended* and *recorded-as-intended*.

[17] For improved efficiency, these encryptions can be combined in a single multi-recipient ElGamal encryption, which can later be split up into different mix-net inputs.

- An important aspect of universal verifiability is eligibility verifiability. It must be ensured that only votes from eligible voters and only one vote per eligible voter is included in the final tally. It seems that this aspect is completely missing in the presented proof.
- It is claimed that the proof shows that the configuration phase "is correct and leaks no sensitive information" and later that the setup is "correct and private". We doubt that this is a correct claim. Specifically, the resulting CMtable contains the encrypted short choice return codes. The proof only shows that an adversary can not distinguish between the correct CMtable and random values. As the short choice return codes are selected uniformly at random by the print office, an adversary is also not able to distinguish the CMtable from random values if the short choice return codes were not encrypted. However, unencrypted short choice return codes directly affect the correctness of the protocol, as an adversary knowing all possible choice return codes in use may have a higher probability in guessing the correct choice return code as the required 0.1% defined in [DraftOEV, Art. 2.11.1]. This may happen particularly in elections with a small electorate and a small amount of voting options.
- The protocol involves relatively complex data flows between untrusted parties (such as Voting Server and Voting Client) and parties trusted as a group (such as CCRs, CCMs, Auditors, and Electoral Board). Especially during the voting and tally phase, the interactions between different parties are substantial and critical. A number of attacks on correctness and vote secrecy have already been found and reported. The proof as presented does not accommodate these interactions. The proof is conducted on an abstraction level which is too high and hence does not fully cope with the complexity of the real protocol. We conclude that the proof provides evidence that certain aspects of the protocol fulfill the security goals but not that the complete protocol satisfies these security goals. We recommend to either revise the proof or to reduce the complexity of interactions between parties in the protocol.

3 Systematic Analysis

To present our findings from Section 2 in a more systematic manner, we took the content of the document "*Mapping VEleS Anhang*" from the Federal Chancellery as a template to discuss all items from the draft OEV annex that are relevant in Scope 1. The following table indicates for each item whether corresponding [DraftOEV] requirements are sufficiently met according to our examination.

2.	Cryptographic protocol requirements for complete verifiability (Art. 5)	
2.1	System participants	
	In Subsection 2.1, we have discussed some problems related to the parties and their roles in the protocol. Most problems are due to protocol referring to an outdated OEV version. The most critical deviation from the system participant list as defined in [DraftOEV, Annex 2.1] is the print office performing several tasks that should clearly be assigned to the (inexistent) setup component.	×
2.2	Communication channels	
	Subsection 2.1 also lists some problems related to the protocol's communication model, which is not fully aligned with the model as defined in [DraftOEV, Annex]. Three print office output channels are clearly not permitted.	×
2.3	Attackers	
2.3.1	As discussed in Subsection 2.4, several cryptographically relevant aspects of the protocol are underspecified. An example is the generation and distribution of the prime mapping table pTable to all involved parties, which is a critical step of the protocol. Without specifying the details of this process, it is impossible to exclude the possibility for an adversary to infiltrate different prime mapping tables, e.g., with the goal of inverting the election outcome.	×
2.3.2	This is a definition of the adversary's capabilities, not a requirement.	–
2.4	Trustworthy and untrustworthy system participants and communication channels	
2.4.1	Requirements met.	✓
2.4.2	This is a definition of trustworthy and untrustworthy parties and channels, not a requirement.	–

2.5	Requirement for the cryptographic protocol: individual verifiability	
	The main problem here is the absence of a mechanism that allows voters to check reliably that no adversary has maliciously cast a vote on the voter's behalf. We have discussed this problem and possible solutions in Subsection 2.3.	×
2.6	Requirement for the cryptographic protocol: universal verifiability	
	An attacker can manipulate the election result by infiltrating different prime mapping tables (see discussion in Subsection 2.4.1). The goal of such an attack is not to modify, suppress, or produce votes, but to change the interpretation of the final tally.	×
2.7	Requirements for the cryptographic protocol: voting secrecy and absence of premature results	
2.7.1	As discussed in Subsection 2.5, we have encountered a problem that may affect vote privacy in the presence of voters with restricted eligibility. This is an intrinsic problem of the protocol design, which is independent of an adversary being present or not. In other words, even the weakest possible adversary breaks vote privacy in such cases.	×
2.7.2	Here the draft OEV seems to be inconsistent. A malicious print office who knows the short choice return codes of all voters can easily breach vote privacy by colluding with a malicious voting server (who learns the short choice return codes of the submitted votes). This is contradictory to the assumption of the print office being fully trustworthy.	?
2.7.3	These requirements are clearly not realistic in any practical setting. It is therefore not a problem of the cryptographic protocol.	?
2.8	Requirement for the cryptographic protocol: effective authentication	
	Requirements met.	✓
2.9	List of trustworthy and untrustworthy system participants	
2.9.1	For soundness of the a proof referred to in Number 2.5	
2.9.1.1	Requirements met.	✓
2.9.1.2	Requirements met.	✓
2.9.2	For soundness of the a proof referred to in Number 2.6	

2.9.2.1	Requirements met.	✓
2.9.2.2	Requirements met.	✓
2.9.3	For preserving voting secrecy and for the absence of premature results in accordance with Number 2.7	
2.9.3.1	Requirements met.	✓
2.9.3.2	Requirements met.	✓
2.9.3.3	Requirements met.	✓
2.9.4	For soundness of the a proof referred to in Number 2.8	
2.9.4.1	Requirements met.	✓
2.9.4.2	Requirements met.	✓
2.10	List of trustworthy and untrustworthy communication channels	
2.10.1	According to [DraftOEV, Annex 2.3.2 and 2.4.2], trustworthy communication channels keep transmitted messages secure (confidential) and authentic, whereas untrustworthy communication channels are potentially under the adversary's full control. Under this premise, we expect to see cryptographic measures to protect untrustworthy channels whenever needed. In most cases, authentication is more critical than confidentiality, but as explained in Subsection 2.4.6, authentication is largely underspecified. Therefore, we can not confirm that corresponding channels are treated as untrustworthy.	×
2.10.2	Requirements met.	✓
2.11	Additional requirements for the soundness of a proof	
2.11.1	In Subsection 2.6, we discussed the scenario where a small electorate and a small amount of voting options implies a very limited number of different short choice return codes over all voters of the given election. In such a setting, a proof built on the indistinguishability between CMtable and a equally long list of random values is not convincing w.r.t. the required $\leqslant 0.1\%$ attack probability, i.e., there must be a problem either in the proof or the protocol.	×

2.11.2	The problem here is the same as discussed above under 2.3.2. If an adversary manages to invert the election result by infiltrating different prime mapping tables, both the effect of the attack and the success probability are 100%.	×
2.11.3	This requirement is not relevant for the protocol.	–
2.12	Functional requirements for the voting process with implications for the cryptographic protocol	
2.12.1	Requirements met.	✓
2.12.2	This requirement is not relevant for the cryptographic protocol.	–
2.12.3	This requirement is not relevant for the cryptographic protocol.	–
2.12.4	This requirement is not relevant for the cryptographic protocol.	–
2.12.5	Requirements met.	✓
2.12.6	Requirements met.	✓
2.12.7	Requirements met.	✓
2.12.8	Requirements met.	✓
2.12.9	Requirements met.	✓
2.12.10	Requirements met.	✓
2.12.11	Requirements met (no voting data imported).	✓
2.12.12	Requirements met (imported data is not confidential).	✓
2.13	Requirements for the definition and description of the cryptographic protocol	
2.13.1	Requirements met.	✓
2.13.2	Subsection 2.4 lists several areas of the protocol specification, in which a too large room for possible interpretations remains. Some of the encountered problems are potential threats to weakening the protocol's security properties.	×
2.14	Proofs of compliance with the cryptographic protocol requirements	

2.14.1	Symbolic Proof: On the publicly available GitLab repository *E-voting Documentation*, the Swiss Post has published various ProVerif models for checking the requirements.[18] Since this is not our area of expertise, we have not evaluated this aspect.	?
2.14.1	Cryptographic Proof: In Subsection 2.6, we have expressed our concern that some of the abstractions of the cryptographic proof seem to be overly simplified.	×
2.14.2	Requirements met.	✓
2.14.3	Requirements met.	✓
4.	Voting process	
4.12	Irrelevant for cryptographic protocol.	–
25.	Quality of the source code and documentation	
25.1	Traceability	
25.1.3	As explained in Subsection 2.1, the given links to the legal requirements have not been updated to the relevant documents [DraftOEV] and [DraftOEV, Annex].	×
25.3	Consistency	
25.2.3	Subsection 2.4 lists several areas, in which the current protocol specification is clearly underspecified.	×

[18] See `https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/tree/master/Symbolic-models`

4 Minor Issues and Typos

Section 3

- Page 14: The algorithm ParamsGen remains largely undefined. From a cryptographic point of view, this is very problematical, because the process of selecting p and q must obey certain rules (their lengths must correspond to the security parameter λ, p must not be close to a power of two, etc.). Without specifying the details of ParamsGen (for example in pseudo-code), all involved parties need to check the quality of the parameters before using them.
- Page 14, Subsection 3.1: A key generation algorithm for the multi-recipient ElGamal encryption scheme is missing.
- Page 14, Subsection 3.1: In the special case $\ell = k$, it would be more consistent to write $pk_\ell^r \cdot m_\ell$ instead of $pk_k^r \cdot m_\ell$.

Section 5

- Page 16, last paragraph of Section 5: The notion of a "weak pseudorandom function" is undefined.

Section 6

- Page 17: Algorithm DeriveKey uses bitLength(p) and byteLength(p) without defining them.
- Page 17: The number of iterations should be justified and related to the security level.

Section 7

- Page 23, second paragraph: $\mathbb{Q}_p \subset \mathbb{Z}_p^*$ rather than $\mathbb{Q}_p < \mathbb{Z}_p^*$
- Page 23, Subsection 7.3.2: The definitions of the parameters of ProveExp and VerifyExp seem to be correct. However, in all applications of ProveExp and VerifyExp, there is a problem with the actual arguments passed to the algorithms, see Pages 60, 64, 66, 67, 68, 69, 71, 74, 90, 100 (at least one unnecessary pair of extra parentheses seems to be present everywhere).
- Page 24, Subsection 7.3.3: In the plaintext equality proof we recommend replacing $(c_0, \overline{c}_0, c_1/\overline{c}_1)$ in st by $(c_0, \overline{c}_0, c_1, \overline{c}_1)$, because otherwise the proof is not unambiguously tied to the two input ciphertexts (c_0, c_1) and $(\overline{c}_0, \overline{c}_1)$.
- Page 24, Subsection 7.3.4: The division $\overline{\mathbf{c}}/\mathbf{m}$ of two vectors $\overline{\mathbf{c}}$ and $\mathbf{m}$ is undefined.

- Page 24, Subsection 7.3.4: $\mathbf{aux}'$ in VerifyDec does not correspond to the $\mathbf{aux}'$ in ProveDec.

Section 8

- Page 25, Subsection 8.1: It is not explained how n and m are selected such that $N = nm$.
- Page 26, Subsection 8.2: The value CRS is undefined.
- Page 27/29/31, Subsection 8.3.2/8.3.3/8.3.5): The computation of the challenges x, y and z does not correspond to the definition in Subsection 8.2. Multiple parentheses are missing.

Section 9

- Page 41, first line of 9.1: Problems cannot be *weaker*, only corresponding hardness assumptions.

Section 10

- Page 45, first line of 10: What exactly is meant by "the system"? Please clarify.
- Page 45, Subsection 10.2: Who exactly is meant by an "external authority"? We recommend using the numbers $1 \ldots, N$ as unique identifiers for the voters, in which case no such an "external authority" is needed.
- Page 45, fifth line of 10.3: The injectiveness of the function encodeVotingOption clearly depends on pTable, i.e., it is not an intrinsic property of the function. Unfortunately, the generation and properties of pTable are undefined.
- Page 45, equation at the bottom: If the product goes over the whole vector, indicating the bounds of the index i is not necessary. Alternatively, the vector can be replaced by its values indexed over i.
- Page 46: In our remarks in Subsection 2.4.7 and in Footnote 16, we have already pointed out that the overall clarity of Section 10.4 should be improved. To the best of our knowledge, there are two different **correctnessID** vectors, one of length n and one of length ψ. The current description is therefore incomplete.
- Page 49, Figure 19: For the data flow of the voting server, it would be simpler to return the values $\mathbf{pk}_{\mathrm{CCR}_j}$ immediately to the CCRs upon receiving them.

Section 11

- Page 49, Figure 19: There is no need to pass the set $\{\mathbf{pk}_{\mathtt{CCR}_j}\}_{j=1}^{m}$ from the print office back to the voting server. The voting server is already in possession of the public keys.
- Page 50, Figure 20: $\mathsf{Enc}_{\mathtt{s}}$ should be called explicitly in an algorithm and should not be part of the data flow. Additionally, the encryption key is not specified.
- Page 52, Figure 21: In addition to $\mathtt{s}_{\mathtt{id}}$ and $\mathtt{SVK}_{\mathtt{id}}$, voters have to provide their identity $\mathtt{VC}_{\mathtt{id}}$ in order to retrieve the keystore $\mathtt{VCks}_{\mathtt{id}}$ from the voting server. The additional round trip between voting client and voting server to retrieve the keystore should also be included in the figure.

Section 12

- Page 58, Step 5 of Algorithm GenVerDat: It should be mentioned that this step is performed for each voter id.
- Page 58, Step 5 of Algorithm GenVerDat: Theoretically, $\mathsf{H}(\mathtt{pCC}_{\mathtt{id},i})$ could be 0, and so could $\mathtt{hpcc}_{\mathtt{id},i}$ be 0 as well. But then the ElGamal encryption in the next step doesn't work. We recommend defining a proper encoding function $\Gamma : \mathbb{N} \rightarrow \mathbb{Q}_p$ such as $\Gamma(x) = (x+1)^2 \bmod p$ and use it whenever needed.
- Page 59, Step 6 of Algorithm GenVerDat: $\mathbf{hpccHash}_{\mathtt{id}}$ is computed for $\forall i \in (i, \ldots, n)$ but index i is completely missing in the computation of the hash value.
- Page 60, Step 2 of Algorithm GenEncLongCodeShares: ConfirmStr is unspecified.
- Page 60, Step 3 and 4 of Algorithm GenEncLongCodeShares: The auxiliary input **aux** of ProveExp is unspecified.
- Page 60, Algorithm CombineEncLongCodeShares: The algorithm parameters should not be index by j. They are vectors of size m.
- Page 61, Step 2 of Algorithm GenCMTable: Index i is missing by the **correctness-ID**.
- Page 62, Step 2 of Algorithm GenCredDat: **KEYseed** is actually a salt and should be different for every voter. We propose to include voter's identity $\mathbf{VC}_{\mathtt{id}}$ in the salt.
- Page 65, Step 3 and 5 of Algorithm CreateVote: Enc and MultiEnc as defined in Section 3 do not expect r, resp. r' as argument.
- Page 65, Step 6 of Algorithm CreateVote: ν should be ρ.
- Page 66, Step 7 of Algorithm CreateVote: **aux** is unspecified.
- Page 66, Algorithm PartialDecryptPCCj: $\mathsf{Logs}_{\mathtt{CCR}_j}$ is missing in the list of parameters. On the other hand, E1 and E2 are unnecessary.

- Page 67, Step 6 of Algorithm PartialDecryptPCCj: **aux** is unspecified.
- Page 68, Step 2 of Algorithm CreateLCCSharej: According to Step 9 in DecryptPCCj the set $\mathsf{L}_{\mathsf{decPCC},j}$ consists of pairs $(\mathtt{vcd}_{\mathtt{id}}, \mathbf{pCC}_{\mathtt{id}})$. Hence, the check $\mathbf{b}_{\mathtt{id}} \in \mathsf{L}_{\mathsf{decPCC},j}$ will always fail. Similar applies to the check $\mathbf{b}_{\mathtt{id}} \in \mathsf{L}_{\mathsf{sentVotes},j}$ and $\mathbf{b}_{\mathtt{id}} \in \mathsf{L}_{\mathsf{confirmed},j}$.
- Page 69, Step 3 of Algorithm CreateLCCSharej: What if the check fails?
- Page 69, Step 7 of Algorithm CreateLCCSharej: Index i is completely missing.
- Page 70, Step 5 of Algorithm ExtractCRC: Dec_s rather than Dec^s
- Page 71, Step 1 of Algorithm CreateLVCCSharej; Wrong reference and the check $\mathbf{b}_{\mathtt{id}} \notin \mathsf{L}_{\mathsf{sentVotes},j}$ always succeeds as the set $\mathsf{L}_{\mathsf{sentVotes},j}$ consists of tuples $(\mathtt{b}_{\mathtt{id}}, \mathbf{pCC}_{\mathtt{id}}, \mathbf{lCC}_{j,\mathtt{id}}, \pi_{\mathsf{exp},j})$ according to Step 11 in CreateLCCSharej.
- Page 72, Step 3 of Algorithm ExtractVCC: Dec_s rather than Dec^s
- Page 76, Step 1 of Algorithm MixDecOnlinej: bb and $\mathsf{L}_{\mathsf{bb},j}$ are neither declared as parameters nor explicitly extracted from the logs.

Section 15

- Page 90, Definition 16: *in* instead of $\in$.
- Page 91, Definition 17: Missing index i in computation of $\mathbf{hpccHash}_{\mathtt{id}}$.

Section 20

- Page 125, second sentence: The logic of the sentence "Since p and q are large primes, quadratic residuosity implies $p = 2q + 1$" is not correct.
- Page 125, Subsection 20.3: We recommend using SHA3 instead of SHA2-256.
- Page 128, Table 8: It is unclear how $1 - 2^{-13}$ and $1 - 2^{-26}$ are derived from the sizes 10^4 and 10^8 of the spaces $\mathcal{C}_{cc}$ and $\mathcal{C}_{vcc}$ (note that $\log_2 10^4$ is slightly bigger than 13).

References

[1] R. Haenni, E. Dubuis, R. E. Koenig, and P. Locher. Process models for universally verifiable elections. In R. Krimmer, M. Volkamer, V. Cortier, R. Goré, M. Hapsara, U. Serdült, and D. Duenas-Cid, editors, *E-Vote-ID'18, 3rd International Joint Conference on Electronic Voting*, LNCS 11143, pages 84–99, Bregenz, Austria, 2018.

[2] R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis. CHVote protocol specification – version 3.1. *IACR Cryptology ePrint Archive*, 2017/325, 2020.