Final Report on Releases 1.4.0 to 1.4.3

Thomas Haines

11th of July, 2024

Summary

We were asked to review version 1.4 of the Swiss Post e-voting system [7] with regards to properties described in the Audit Concept [5], specifically Scope 1 and parts of Scope 2 (d,e,f).

Significant issues—we raised in past reports—have been mostly addressed but we continue to discover new problems which violate the privacy and integrity objectives of the system within the threat model; one of the issues we found is the result of recent changes (Sec. 4.1.1) while another have been present in the system for several years (Sec. 4.2).

There is a growing divergence between the quality of the implementation, which is maturing, and the quality of the specification and proofs. Since current improvements to the specification and proofs will likely become partly obsolete when measures like A.05 are implemented, we can see the logic in not focusing on theses tasks. However, that does leave the system in a non-compliant state with regards to the Ordinance requirements on clear specifications and proven security.

This review was initially encumbered by the change to encrypt the verifier payload; while this issue has now been resolved (through tooling) it was a significant impediment to audibility without any consultation we are aware of; in future, we would encourage any changes which significantly decrease auditability to be discussed outside of an examination period.

We will discuss the status of issues we have raised in previous reports in Section 1 and discuss the status of the computational proof in Section 2. In Sections 3 and 4 we will discuss new issues emerging in Scope 1 and Scope 2 respectively. In the appendices of this report we reflect the status the measures A.10 (App. A), A.20 (App. B), and the various requirements within scope 2 (App. C); the notable points from these examinations appear in the body report.

Significant Findings in this Round of Examination

- The intended privacy of the system could be violated by being getting an honest control component to shuffle and decrypt a subset of the

confirmed votes in a ballot box; fortunately, this issue would have been detected if it had been exploited and has now been patched. For full details see Section 4.1.

- An adversary could remove entire ballot boxes from the result, which the honest control component believes should be tallied, without detection. For full details see Section 4.2.
- The privacy proof does not reduce properly to security of the encryption scheme (IND-CPA). For full details see Section 2.1.

High Level Design and Presentation

Before going into details on specific issues we wish to make a high level observation about how the system and verifier specifications are structured. Recall that one of the OEV requirements for cryptographic protocol is:

> 2.13.2, 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.

This requirement while taking only one sentence to state is complicated to fulfill in a system as complex as the Swiss Post e-voting system; it is not currently fulfilled.

To achieve this requirement in a reasonable space requires carefully chosen layers of abstractions in a specification. At present neither the system or verifier specifications have the right kind of abstractions. One positive example of use of abstraction layers is the presentation of digital signatures in the system specification which do not clutter the main presentation but are given in detail in Section 7 and particularly Table 15.

System Specification

The system specification covers a lot of the protocol in reasonable detail including the algorithms and digital signatures. However, the description of how the components should manage state and what protocol flows are valid is communicated principally through the sequence diagrams and stateful lists. These leaves open to interpretation many questions, a few of the security related ones we highlight in Sec. 3.5.

Verifier Specification

At present the verifier specification treats all checks required on the same level of abstraction and so cryptographic checks are described at the same

level as validations of ensuring all the data has been received or that file names are correct. This not ideal in that presentation of the core protocol checks are cluttered by many which exist at a much lower level of abstraction than exists in the system specification. Moreover, many of the non-core protocols checks are underspecified and consist principal of text rather than pseudocode.

Contents

1 Update on issues flagged previously

In this section we will recap outstanding issues raised in previous reports and provide an update on the current status. However, we leave discussion of the computational proof to the next section (Sec. 2).

1.1 Auditor Instructions

In our most recent report [3] we stated that

> We think that is is crucial to amend the documentation with a detailed description of the use of the verifier by the auditors at the operational level. The verifier's sufficiency for universal verifiability is quite sensitive to how it is used. The voting system, like any engineered system, exhibits a tolerance between the specification, implementation, and operation; in other words, while the implementation and operation are largely the logical consequence of the specification there is a degree of imprecision, as seen in the code and the documented procedures. The level of variation in the voting system is such that without knowing ahead of time how the verifier will be operated there is an unacceptable risk that it will not ensure the universal verifiability of the system. Until this is addressed we believe that the systems should considered non-compliant with Art.3.a and Art.3.c of the OEV."

– *Status* –

Post has made some minor changes to the manual checks the auditor is supposed to perform but there has been no significant progress on this issue.

Within scope 2, we would like a step-by-step instruction list of everything the auditor (not the technical aid) is supposed to do; we realise such a document would be long and hints about this are spread across many documents but consolidating these would be helpful; for example, the auditor is supposed to check the results confirmed by the technical aid match the announced results (Verifier Spec Sec. 2.4) but for Post's verifier implementation we are ensure how to see the results in the verifier (in conversation with Post, it has been made clear that this is supposed to be done with a separate program).

Prior to Version 1.4.3, looking at the algorithm describing the manual checks, included here as Fig. 1, only for operation 7 is it clear how this check should be performed. For all the others it is unclear what should be done. We will provide an example ambiguity for each operation below:

1. Checking the (context dataset's) hash is relatively straightforward but the context dataset's hash from the `VerifyConfigPhase` is not an input to this algorithm.

Verification 0.01 ManualChecksByAuditors

Context:

A verifier's execution in the VerifyConfigPhase or VerifyTally.

Input:

The verifier's report(s) indicating the domain-specific parameters.
The protocol participants' certificates—received via an out-of-band channel (section 2.3).

Operation:

1: When executing VerifyTally, check that the context dataset's hash equals the one from the VerifyConfigPhase.
2: Check that the certificates' fingerprints match the expected ones.
3: Check that the name and date of the election event correspond to the expected ones.
4: Check that the number of elections and votes corresponds to the expected one.
5: Check that the number of productive and test ballot boxes corresponds to the expected one.
6: Check that the number of real and test voters corresponds to the expected one.
7: Check that all expected verifications executed successfully.

▷ See crypto primitives specification

Output:

$\top$ if the verification succeeds, $\bot$ otherwise.

Figure 1: ManualChecksByAuditors - Verifier Specification Version 1.5.1

2. One assumes this checks the fingerprints of the certificates used in the execution against the certificates received out-of-band; however, this is not clear. It is also odd, that the out-of-band certificates are considered untrusted (inputs) whereas the execution (and therefore by extension the data used in the execution) is trusted (context).

3. It seems reasonable that the name and date of the election should be known to the auditors but it is not clearly given as an input to this algorithm.

4. Again, it seems reasonable to assume that the auditors know the number of elections (by which we assume is meant groups of voters with different voting options) but this is not given as an input to the algorithm.

5. It is less clear how the auditors should know how many production ballot boxes there should unless this is supposed to follow directly from the earlier data.

6. The data provided by the verifier makes it easy to check the total number of voters and cast votes but not to see how this breaks down to voters and cast votes in each election (verification card set/ballot box). This seems odd since surely what we wish to ensure is that each election has the correct number of voters not that the total number of votes is correct.

– *Status* –

In version 1.4.3, Post addressed most of the issues noted above with the algorithm with the exception of the last. Moreover, it remains the case that the verification that the announced election result matches the verified election result is underspecified (it does not appear in the manual checks algorithm) and may allow an unacceptably high probability of undetected variation.

In very recent discussions with Post it was clear that they intend, and moreover believe to be the case in practice, that the result which is verified becomes the published result; this would eliminate the need to check the verification with the published result since no such result exists at the point of verification. There is a slight resulting increase in trust on the auditors since there is no result to check their announced result with but this is allowed within the threat model. In summary, Post's position on the lack of need for published result verification makes sense to us but would need to be clearly communicated in the system specification. Moreover, a method should added for the auditors to check that the result published matches the result they verified; for example, by having the verifier display the hash values of the XML tally files.

1.2 Voter Authentication

Our impression is that the voter authentication introduced by Post has not been well received by the mandated experts. In our previous report [3] we wrote:

> Overall, the voter authentication process appears to remain weak, and may be one of the main practical weaknesses of the voting system in general. However, to the best of our understanding, this is not related to any specific issue that could be overcome within the proposed voting system, but rather to the lack of availability of a remote authentication infrastructure for Swiss voters.

Other reviewers had comments along the same vein; For example, the reviewers from BFH (Sec. 3.1 of [1]) criticised the new authentication mechanism as creating usability problems without any clear security advantage.

– *Status* –

Post has made changes to ease the usability problems but the authenication mechanism remains in place.

1.3 Status of Scope 2 Requirements

We have provided summaries of the statuses of the Scope 2 (d, e, and f) requirements in Appendix C. Many of the requirements are clearly satisfied

and most of the unsatisfied requirements are so because of missing details in the specification.

Interestingly across the various releases of the system, at the implementation level there have been notable less confirmed vulnerabilities affecting individual verifiability than either privacy or universal verifiability. We suspect this reflects both the inherent brittleness of privacy compared to the other two properties and the fact that the setup component is trusted for individual verifiability but not universal verifiability. This suggests that measure A.05 which will reduce reliance on the setup component will, if done well, also reduce instances of vulnerabilities affecting universal verifiability.

1.4 Controls on Micro-service Execution

In [3], we noted that the micro-services in the control components from the config phase might still accept input in the later phases.

– *Status* –

We are pleased to see that the micro-services for the config phase, with the exception `KeyGenerationProcessor`, now include explicit checks that the election is in the expected state. We expect based on the database rules that such checks are unnecessary for the `KeyGenerationProcessor` though it would be nicer if this were more readily verified.

1.5 Consensus and Universal Verifiability

In Section 5.2 of our most recent report [3] we highlighted various consensus issues within the system under the threat model of Universal Verifiability which we will provide updates on below.

1.5.1 Filtering of Invalid Selections

We noted that system was not compliant with the Ordinance because it did not ensure that tallied ballots were valid within the threat model of universal verifiability.

– *Status* –

Post has added various checks to the verifier to address this issue and it should be resolved. The specific issue of a party list without candidates is reported to be enforced by the way eCH-0110 files are generated, the details of which were not transparent to us without further discussions with Post.

1.5.2 Verification of Vote Numbers in Control Components

In our report of July 2023 [3] (Sec. 5.2.3), we discussed how the `control components` receive a view of the number of eligible voters but don't use that information. We recommended:

- A check in the VerifyConfig phase of the verifier which checks the consistency of the election configuration received by the control components.
- A check in the `GenEncLongCodeSharesProcessor` that the number of voting cards processed is less than or equal to the number of eligible voters.

– *Status* –

- As of Version 1.4.3 the consistency of the election event configuration is checked as part of the `VerifySchnorrProofs`, see Section 4.2 for more details of why this was needed.
- Post has added a check to ensure the number of voting cards processed by `GenEncLongCodeSharesProcessor` does not exceed the number of eligible voters.

1.5.3 Missing Verification Checks in Specification

Every major definition of verifiability in the literature assumes a clear correspondence between encrypted ballot and plaintext vote which is independent of any trust assumption. In Section 5.2 of our most recent report [3], we discussed the (surmountable) difficult of realising this in a deployed system and stated we believed the repeated assumed correspondence emphasised that this was an important detail to take care of. In summary, we argued that the following subproperty is crucial to any reasonable interpretation of the universal verifiability requirement.

> - The system must include sufficient verification checks to ensure a consistent view of the votes between the `control components` and `verifier`.
>
> A consistent view of the votes means both the group elements making up the ciphertexts and all information required to give those a definitive meaning in the language of the result. Without a consistent view between the `control components` and the `verifier` the language of "the votes" used in the OEV is simply undefined.

– *Status* –

We believe this issue is now resolved. Post has amended the verifier specification to ensure that for each verification card set, for which at least one ballot is submitted, the verifier and control components agree on all information giving meaning to the encrypted ballots.

2 Analysis of Computational Proof

In this section will provide an update on various previously flagged and new issues in the computational proof.

2.1 Contradiction in the Privacy Proof

The basic concept of a (security) reduction is to show that an adversary who can solve problem A can be used to solve problem B. This will commonly require some additional computation but this computation must be much easier than problem B otherwise the reduction is without meaning.

Speaking informally, the privacy reduction (Sec. 18.3) shows that an adversary able to break privacy must be able break one of a number of problems which are believed to be hard. From this we can deduce that privacy should hold. One of these problems is the IND-CPA security of the ElGamal ciphertexts.

The issue at present is that the challenger playing the honest control component explicitly uses it's knowledge of the election secret key. Part of the implicit reasoning in game 14 is that the IND-CPA challenger is providing a public key which is being used as the honest control component's key share. So at this point, the honest control component no longer knows the secret key and cannot do any of tasks which it has been told to do.

To be clear, we believe the system as described does satisfy the definition of privacy given. However, the current proof does not demonstrate this.

2.1.1 Sketch of an IND-CPA Game Hope

In an attempt to clarify the point above we will give an attempted game hope for a much simpler system.

While IND-CPA is not formally defined in the "Computational Proof of Complete Verifiability and Privacy" recall that is commonly defined along the following lines.

Description of experiment	$Exp_{\mathcal{A}}^{ind-cpa-b}()$
Keys are generated	$(pk, sk) \leftarrow\!\!\$\ \texttt{KeyGen}()$
Adversary chooses two vectors of messages	$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}(pk)$
One of the two vectors of messages is encrypted	$\mathbf{c} \leftarrow\!\!\$\ \texttt{GetCiphertext}(\mathbf{m}_b, pk)$
The adversary guesses which one	$b \leftarrow\!\!\$\ \mathcal{A}(\mathbf{c})$

Note that we have used slightly simpler notation by using `KeyGen` for both parameter and key generation and made the sampling of randomness an internal part of `GetCiphertext`.

In a privacy game a reduction to IND-CPA has the challenger (in the privacy game) play the adversary (to the IND-CPA game). That is the challenger gets given the public key, gets to choose the vectors of ciphertexts, and must use the actions of the adversary to the privacy game to guess which of the two vectors of ciphertexts was encrypted.

Consider the following privacy game for a much simpler e-voting system.

Description of experiment	$Exp_{\mathcal{A}}^{privacy-b}()$
Keys are generated	$(pk, sk) \leftarrow\!\!\$\ \texttt{KeyGen}()$
Adversary chooses two vectors of honest votes	$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}(pk)$
and some encrypted votes belonging to dishonest voters	$\mathbf{c}_d \leftarrow\!\!\$\ \mathcal{A}(pk)$
One of the two vectors of votes is encrypted	$\mathbf{c} \leftarrow\!\!\$\ \texttt{GetCiphertext}(\mathbf{m}_b, pk)$
Honest and dishonest votes are shuffled	$\mathbf{c}_m \leftarrow\!\!\$\ \texttt{Shuffle}(\mathbf{c}\|\|\mathbf{c}_d, pk)$
Votes are decrypted	$\mathbf{m}_d \leftarrow\!\!\$\ \texttt{Decrypt}(\mathbf{c}_m, sk)$
The adversary guesses which one	$b \leftarrow\!\!\$\ \mathcal{A}(\mathbf{m}_d)$

An immediate attempt (which will not succeed) to reduce IND-CPA would look like this.

Description of experiment	$Exp_{\mathcal{A}}^{privacy}()$
Get the public key from the IND-CPA challenger	$pk \leftarrow\!\!\$\ \mathcal{C}_{\texttt{IND-CPA}}$
Adversary chooses two vectors of votes	$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}(pk)$
and some encrypted votes belonging to dishonest voters	$\mathbf{c}_d \leftarrow\!\!\$\ \mathcal{A}(pk)$
One of the two vectors of votes is encrypted by the IND-CPA	$\mathbf{c} \leftarrow\!\!\$\ \mathcal{C}_{\texttt{IND-CPA}}(\mathbf{m}_0, \mathbf{m}_1)$
Honest and dishonest votes are shuffled	$\mathbf{c}_m \leftarrow\!\!\$\ \texttt{Shuffle}(\mathbf{c}\|\|\mathbf{c}_d, pk)$
???	$\mathbf{m}_d \leftarrow\!\!\$\ ???$
The adversary guesses which one	$b \leftarrow\!\!\$\ \mathcal{A}(\mathbf{m}_d)$

The updates on step 1 and 4 are straightforward by asking the IND-CPA challenger for the public key and asking the challenger to choose which of the two vectors of ciphertexts to encrypt. However, step 6 has no possible solution for this simple system. If only honest votes where cast then, given the restriction that the two possible sets of votes are equal, we could return these votes (which were seen in plaintext at step 2) in a random order being confident that this is a correct simulation; however, since the adversary has added dishonest votes we need to know the plaintext value of the votes without knowing the decryption key. This is normally achieved by having the adversary submit a proof of knowledge of plaintext which we extract.

This simple example is intended to highlight the fact that a reduction to IND-CPA requires the privacy challenger to operate without knowledge of the secret key. In a more complicated protocol like the Swiss Post e-voting system the proof is much more complicated but still faces this basic constraint.

2.1.2 Usage of Unbounded Extractor

The reduction makes use of an unbounded extractor. This extractor is used on ciphertexts which are given by the IND-CPA challenger; it is unclear if the adversary is actually solving the IND-CPA problem or the unbounded extractor. It would be much cleaner to avoid any use of an unbounded extractor in the privacy proof.

2.1.3 Status

We understand that Post intends to address these issues after the next relevant protocol changes.

2.2 Usage of IdealSetupVoting

Lemma 1 looks very plausible but the usage of this lemma in later proofs is problematic.

The issue is that the adversary in the later games receives more information from setup than the adversary in Fig. 22 does. For example some of voting cards. This means the later game hops are not clearly valid since this more knowledgeable adversary could possibly detect the difference; we believe that the extra information does not actually allow the adversary to detect the change but Lemma 1 should prove this.

We recommend changing Fig. 22 to give the adversary all the information it will see in later usages of Lemma 1, and update the proof of Lemma 1 accordingly.

– *Status* –

We understand that Post intends to address these issues after the next relevant protocol changes.

2.3 16.5 AEAD,IND-CPA hop

Game hop `sai.3` is invalid because it includes randomising voting options for which the adversary may know the secret key.

The adversary will learn the secret key for all voting options on the ballot it submits, to randomise without detection the challenger needs to randomises only for those voting options which don't match the cast ballot. This probably requires guessing what ballot the adversary will cast which would need to be added to the bounds in Theorem 1.

We recommend only randomising those ciphertexts where the adversary does not learn the secret key.

– *Status* –

We understand that Post intends to address these issues after the next relevant protocol changes.

2.4 Definition of Correct Tally

Figure 34 of the computation proof defining the correct tally property says (on line 7) that the adversary loses if the plaintexts don't process properly even if `VerifyTally` succeeds (line 4). Why should an adversary not win in this case?

Given that `VerifyTally` includes checking that `ProcessPlaintexts` succeeds this restriction seems redundant.

– *Status* –

We understand that Post intends to address these issues after the next relevant protocol changes.

2.5 Alignment of Security Model in Proofs with Ordinance

Recall that requirement 2.14.1 of the Ordinance states:

> 2.14.1 A symbolic and a cryptographic proof of compliance must demonstrate that the cryptographic protocol meets the requirements in Numbers 2.1–2.12.

The improvements to the proof document have been significant but there are outstanding issues:

2.5.1 Role of auditors

In Section 2.1.1 of our February 2023 report [4], we noted that the auditors use during the setup phase is problematic and their use to ensure privacy does not align with the trust model in the Ordinance.

– *Status* –

This issue is covered by measure A.22 [6] and is scheduled resolution by the end of 2026.

2.5.2 Roles of tally control component and electoral board

In Section 2.1.2 of our February 2023 report [4], we wrote

> By modeling the Tally Control Component, together with the Electoral Board, as a single Control Component the analysis only considers what happens if all these parties are honest and not what happens when some of these parties misbehave but others don't.

– *Status* –

Post has weakened the assumption to consider only a single member of the electoral board as honest.

2.5.3 Alignment of Protocol Participants

The Annex 2 to the OEV describes the security requirements for the system. Establishing a clear correspondence between participants in 2.1 and the Post's cryptographic protocol is essential to justifying that the protocol meets the requirements. Section 2.1.2 of Cryptographic Protocol document aims to make this correspondence clear but has two main issues:

1. Two correspondences are given which allows a degree of equivocation in the claims about how the security properties are satisfied.
2. The OEV allows for multiple groups of control components. Reading between the lines it appears that Post wishes to have three groups but that this is not made explicit.

Our understanding is that the proof essentially assumes three groups of control components:

Setup CC Group: Online Control Components

Tally CC Group: Online Control Components + Tally Control Component

EB CC Group: Electoral Board

– *Status* –

After discussion with Post on this point there are no immediate plans for changes. We remain of the view that a system which conformed to the requirements of the ordinance would be able to provide a mapping of participants which was consistent. That the current solution by Post is unable to do this without creating undue confusion is a problem. This does not need to be immediately resolved but should be taken into account when measures like A.05 (where the protocol will be significantly re-engineered) are undertaken.

The involvement of the Electoral Board seems to be important to some of the procedural requirements but unideal for the cryptographic proofs. A future change as part of A.05 should consider changing the protocol to allow the Electoral Board to be untrusted for the cryptographic proofs.

3 New Examinations and Issues: Scope 1

In this section we will comment on major protocol changes and issues within scope 1.

3.1 Encoding of Votes in CreateVote

As of version 1.4.0 of the system specification delegates the encoding of votes to `CreateVote`. That is, it now takes a string encoding the vote option (see Sec. 3.5.2 of the system spec for format) and uses the prime mapping table to convert this to a group element. It is good to see the specification using a representation of the vote which is less tied to the internals of the system as input.

3.2 Inconsistent Naming of Setup/Configuration Verification

It is unhelpful that the verification before voting commences is referred to as `VerifyConfigPhase` in system specification but predominantly as some variation of `Setup Verification` in the verifier specification and implementation. It would be simpler to either use the word configuration or setup but not both.

– *Status* –

We understand that Post intends to address this in a future release.

3.3 Undefined notation in completeness checks

The completeness checks use the notation $* in the specific instances:

- $j
- $vcs
- $chunckId
- $bb

This notation is currently undefined in the document and it is unclear, particular in the latter three cases, how to interpret it. We can assume that $j will be the set of control components but if the other three are to be the sets of verification card set, chunks ids, and ballot boxes, how does the verifier know what these sets are?

– *Status* –

This was addressed in version 1.4.3 with the addition of a paragraph to the verifier specification detailing how the notation is to be interpreted.

3.4 Usage of Algorithms and Chunking

While Table 7 does list how often each algorithm is executed this information could be profitable reiterated when introducing each algorithm.

The implementation, however, deviates from the specification by implementing chucking; this renders the information in Table 7 incorrect with regards to the implementation. For example, the implementation calls Alg. 4.7 once for each chunk making up the verification cards sets. This is an important factor in the security of the system since the sorting of the `CMtable` is done inside the algorithm. For clarity we believe that sorting `CMtable` on a per voter basis should be fine for security and consequently sorting per chunck should also be secure in practice but specification should be clear on this point.

Given the possible security implications of chunking a greater explanation is required as to why the implementation has the same effect as the specification. Alternatively, the specification could be updated to reflect the reality of how these algorithms are called.

– *Status* –

We understand that Post intends to address this at some point but we are not aware of any details on either the planned changes or the timeline.

3.5 Ambiguities in the System Specification

In the summary of this report we noted that the system specification does not clearly specify the overall protocol. Below we give several examples of ambiguities arising in the specification.

- The e-voting system, as described in system specification, consists of five main protocols (`SetupVoting` Fig. 6, `SetupTally` Fig. 7, `SendVote` Fig. 9, `ConfirmVote` Fig. 10, and `Tally` Fig. 11 and 12). However, how these protocols relate in terms of concurrency and even order is unclear from the system specification. The naive interpretation would be that each protocol is executed once in order but this does not work.

  `SetupVoting` and `SetupTally` both occur once per election event, whereas `SendVote` and `ConfirmVote` are intended to execute many times; given the setup occurs once per election event one would assume that `Tally` would be the same especially because `VerifyTally` as described in the verifier specification is described in this way; however, by looking at the details of the algorithms and information sent it becomes clear that the `Tally` protocol (Fig. 11) does not tally the election but rather a particular ballot box. To summarise, the current condition of Fig. 11 is inconsistent and should probably be updated to show the tallying of the whole election event not just a single ballot box.

In short the description of the e-voting protocol as a whole is missing from the system specification; it is unclear how the various presented protocols as supposed to be composed into this protocol. This should be fairly easy to fix but if it is to match the implementation will require introducing notation to denote parts of the protocol that are allowed to execute concurrently and parts of the protocol that are allowed to execute multiple times.

- In several cases it is currently unclear how the component is supposed to know what context/input to use with the algorithm or alternatively the obvious interpretation is inconsistent with the implementation. We give a few examples below:
  - Fig 6. the verification card ids `vc` are sent as a single item rather than per verification card set, how does the control component know which ids belong to each card set? does it use the same set of ids with each card set? This also applies to $\mathtt{c}_{\mathtt{pCC}}$ and $\mathtt{c}_{\mathtt{ck}}$. The allow list $\mathtt{L}_{\mathtt{pCC}}$ is also sent as a single item does this mean despite being generated per verification card set these are merged and used as a single global item for that election event? The information sent from the `CCR`s is also not shown as clustered by verification card set.
  - The vote confirmation allow list $\mathtt{L}_{\mathtt{lVCC}}$ is shown as a single item implying the various allow lists produced by the `GenCMTable` have been merged.
  - In `VerifyBallotCCR` how does the `CCR` know what election event, verification card set, and hence public keys and primes mapping tables to use? Section 1.5.1 says these are to be inferred but there is no guarantee that the verification card ids are unique. (Verification 3.12 `VerifyVerificationCardIdsConsistency` checks the uniqueness within verification card sets but no on any more global level.)
- For various stateful lists and maps it is unclear if these are supposed to be global variables or specific to an individual election event or verification card set.

4 New Examinations and Issues: Scope 2

In this section we will comment on major protocol changes and issues within scope 2.

The tight timelines for our examinations continue to be made more challenging by the raw state of the released materials; for example, missing details in the released code and documentation, and code or scripts which do not execute successful. The main examples of this in the current round of examination where:

- The provided example verifier datasets did not initially include the password required to use them;
- tooling was not initially provided to encrypt new payloads to test the verifier;
- the verifier (for which the instructions are provided for a windows machine) would not accept payloads zipped on windows before version 1.4.2;
- in version 1.4.1 of evoting-e2e-dev, one of the provided build scripts did not execute successfully.

It appears that evoting-e2e-dev is still bugged in version 1.4.2 but we bypassed the issue; one of the required files for the build script is permissioned so the default user cannot access it.

4.1 Restrictions on Micro-services execution in vote and tally

We had previously analysed the restriction on Micro-service execution during `setup` and it was therefore natural to extend this analysis to the `vote` and `tally` phases. The restrictions all seemed appropriate with one notable exception.

4.1.1 Lack of restrications on GetMixnetInitialCiphertextsProcessor

The apparent lack of restrictions on calling the `GetMixnetInitialCiphertextsProcessor` and slightly odd behaviour of the `MixDecryptMessagingService` allowed an adversary to break the privacy of the e-voting system. That is an adversary, operating within the threat model of privacy, learned more about how certain voters voted then it would learn from seeing a random permutation of all confirmed votes in plaintext .

Attack sketch:

Step 1: The adversary calls the `GetMixnetInitialCiphertextsProcessor` of the honest control component at some while voting is open. This triggers the control component to generate it's hash of confirmed votes at this point though it continues to receive and confirm votes after this point. It also saves its view of confirmed votes at this point in the `mixnetInitialCiphertextsService`.

Step 2: After the voting period is over the the adversary calls the `MixDecryptMessagingService`. This service, rather than taking all confirmed votes at that point, looks up in the votes saved previously in the `mixnetInitialCiphertextsService`, it then shuffles and decrypts these votes.

Step 3: The adversary receives the shuffled decryptions of the confirmed votes at the point `GetMixnetInitialCiphertextsProcessor` was called; this is a smaller anonymity set than the full set of votes (for that ballot box) confirmed during the election.

Details: `GetMixnetInitialCiphertextsProcessor`, will performing an impressive array of consistency checks, does not check that it is actually operating in the tally phase. By contrast the `MixDecryptService` has a method called `validateMixIsAllowed` which checks precisely this.

Moreover, `GetMixnetInitialCiphertextsProcessor` does not update any of the underlying storage services to warn them that it has begun tallying. Again, in contrast the `MixDecOnlineAlgorithm` tells that ballot box service that the box has been mixed.

The `GetMixnetInitialCiphertextsProcessor` saves the confirmed votes in the `mixnetInitialCiphertextsService`.

While the `MixDecryptMessagingService`, which handles shuffling and decryption, does perform both an impressive array of consistency checks and checks that it is operating in the tally phase, it performs its mix, decryption, and verification with regards to the confirmed votes in `mixnetInitialCiphertextsService` (saved from the `GetMixnetInitialCiphertextsProcessor`) rather than by looking at the ballot boxed themselves.

Fortunately, after performing the mixing and decryption, `MixDecryptMessagingService` does ask the ballot box service which votes were confirmed and sends these on to the tally control component and verifier; if it had sent the contents of `mixnetInitialCiphertextsService` instead this would also be an attack on universal verifiability, by dropping those ballot confirmed after the `GetMixnetInitialCiphertextsProcessor` was called.

– *Status* –

This was patched in version 1.4.1.

4.2 Consensus on Verification Card Sets

The honest control component may have a different view of the number of verification card sets and number of voters within these sets than the auditor.

The issue with both the verification cards and verification card sets is that the auditor is not given a single payload summarising the control components view but rather many different payloads. If an adversary drops some of these payloads in transit then the auditor has no specific way of knowing this.

Consequently, the control component may believe there are more verification card sets than the auditor (undetected because the corresponding payloads have been dropped); the control component may also believe there are more verification card ids in a known verification card set (undetected because the corresponding chuck payloads have been dropped).

We believe this is violation of universal verifiability, specifically the first clause since the dropped ballots have been registered as cast in conformity with the system.

> 2.6 Requirement for the cryptographic protocol: universal verifiability
>
> The auditors receive a proof in accordance with Article 5 paragraph 3 letter a in conjunction with Article 6 letters a and c to confirm that no attacker:
>
> – after the votes were registered as cast in conformity with the system, has altered or misappropriated any partial votes before the result was determined;
>
> – has inserted any votes or partial votes not cast in conformity with the system which were taken into account in determining the result.

There are restrictions on how the adversary can exploit these inconsistencies but ruling out that none of the possibilities affect universal verifiability unnecessarily complicates the analysis of the system.

– *Status* –

This issue was patched in version 1.4.3 through the inclusion of a hash produced by `GetHashElectionEventContext` as auxiliary data in the (key generation) zero-knowledge proofs generated by the Setup and Online Control Components. This ensures that the honest Online Control Component's view of the election event context is consistent with that of the verifier, which resolves the issues discussed above.

4.3 Aim of verifyCorrectMappingInAllVerificationCardSets in VerifyPrimesMappingTableConsistencyAlgorithm

The method `verifyCorrectMappingInAllVerificationCardSets` in the `VerifyPrimesMappingTableConsistencyAlgorithm` claims to

> Verifies that the same actual voting option has the same encoded voting option, that the same actual voting option has the same semantic information and that the same actual voting option has the same correctness information in all PrimesMappingTables.

However, the the algorithm seems only to check the uniques of the encoded voting options. This is not sufficient to ensure any of the claimed properties.

– *Status* –

This issue was fixed in version 1.4.1

4.4 Lack of Authenticity and Consistency Checks in VerifyTally

Note: We discuss the following issues in the contest of release 1.5.0.0 of the verifier but a similar issue may have been present prior to this release.

As of 1.5.0.0 the auditor will receive three zip files for verification:

- dataset-CONTEXT-*.zip
- dataset-SETUP-*.zip
- dataset-TALLY-*.zip

The first is used in both verification executions, setup and tally, and while the second is only used during `VerifyConfigPhase` and third is only used during `VerifyTallyPhase`.

The issue here concerns if the files which are used in both `VerifyConfigPhase` and `VerifyTallyPhase` are the same between each execution; if they are not the files in `VerifyTallyPhase` may be inconsistent and unauthenticated.

Post had argued that the auditor checks the consistency of dataset-CONTEXT-*.zip, by checking the hash of the file, which ensures the files are the same. This seems to assume that the files which are used in the both phases are contained in dataset-CONTEXT-*.zip rather than the other two zips.

It appears from the `VerifierProcessor` that `electionEventContextPayload` and `configuration-anonymized` are required to be in dataset-CONTEXT-*.zip, but we cannot see any constraint which would require the following files to be in dataset-CONTEXT-*.zip:

- setupComponentPublicKeysPayload

- controlComponentPublicKeysPayload.j
- setupComponentTallyDataPayload (for each card set)

We have asked Post if any such constraints exists since with the lack tooling, we were unable to practically test this ourselves. Post informed that no such constraints existed; in summary, the files (listed above) claiming to be from the setup component during `VerifyTallyPhase` are neither authenticated nor checked for consistency in version 1.4.0.

– *Status* –

This was addressed in version 1.4.1.

4.5 Making the behavior of CharsetEncoder explicit

The security of the hash functions depends on the behavior of the `CharsetEncoder` specifically the `CodingErrorAction`, both replace and ignore would break collision resistance, but the default behavior, report, is secure so this is probably not an issue.

We have asked Post to consider updating `ConversionsInternal` to explicitly require report.

– *Status* –

This recommendation was implemented in version 1.4.3.

4.6 Ciphertext Agreement before Tally

As of version 1.4.0, the control components expect to receive a hash of the ballot boxes from all other control components before they perform their own decryption and shuffling. This prevents partial shuffles and decryptions which would later be aborted due to a disagreement about the ballot boxes.

We have verified that the `MixDecryptMessagingService` receives payloads containing a hash of the ballot box with four different claimed senders. It checks that each of these payloads is signed by the claimed party and that the payloads agree with regards to the hash. This seems to provide the desired functionality.

4.7 CHANGE_CONTROL_IDs and the Ballot_Box table

We had some discussion with Post about the usage of `CHANGE_CONTROL_ID`s to enforce security properties. As a result of these discussions a constraint on `CHANGE_CONTROL_ID` was added to the `Ballot_Box` table.

4.8 Transition from RabbitMQ to Artemis

On a minor note the Architecture document still uses the term RabbitMQ on pages 39 and 74 where we assume this should read Artemis.

References

[1] Rolf Haenni, Reto E. Koenig, Philipp Locher, and Eric Dubuis. Re-examination of the swiss post internet voting system. https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/Examination_reports_August2023/Scopes%201%20and%202%20Final%20Report%20BFH%2030.06.2023.pdf.download.pdf/Scopes%201%20and%202%20Final%20Report%20BFH%2030.06.2023.pdf, June 2023.

[2] Thomas Haines, Olivier Pereira, and Vanessa Teague. Addendum to report on the swiss post e-voting system. https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/Examination_Reports_March2023/Scopes%201%20and%202%20Final%20Report%20Addendum%201%20Thomas%20Haines,%20Olivier%20Pereira,%20Vanessa%20Teague%2021.11.2022.pdf.download.pdf/Scopes%201%20and%202%20Final%20Report%20Addendum%201%20Thomas%20Haines,%20Olivier%20Pereira,%20Vanessa%20Teague%2021.11.2022.pdf, November 2022.

[3] Thomas Haines, Olivier Pereira, and Vanessa Teague. Remarks on selected elements of the swiss post e-voting system versions 1.2.3, 1.3, and 1.3.1 final version. https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/Examination_reports_August2023/Scopes%201%20and%202%20Final%20Report%20Thomas%20Haines,%20Olivier%20Pereira,%20Vanessa%20Teague%2031.07.2023.pdf.download.pdf/Scopes%201%20and%202%20Final%20Report%20Thomas%20Haines,%20Olivier%20Pereira,%20Vanessa%20Teague%2031.07.2023.pdf, July 2023.

[4] Thomas Haines, Olivier Pereira, and Vanessa Teague. Second addendum to report on the swiss post e-voting system. https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/Examination_Reports_March2023/Scopes%201%20and%202%20Final%20Report%20Addendum%202%20Thomas%20Haines,%20Olivier%20Pereira,%20Vanessa%20Teague%2013.02.2023.pdf.download.pdf, February 2023.

[5] Swiss Federal Chancellery. Audit concept v1.5. Available from https://www.bk.admin.ch/dam/bk/de/dokumente/pore/Vote--lectronique/Audit%20concept%20v1.5.pdf.download.pdf, September 2022.

[6] Swiss Federal Chancellery. Catalogue of measures by the confederation and cantons. Available from https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/E-voting%20Catalogue%20of%

20measures%20by%20the%20Confederation%20and%20cantons,%204%20August%202023.pdf.download.pdf, August 2022.

[7] Swiss Post. Swiss post voting system. https://evoting-community.post.ch/, 2021.

A Analysis of A.10 (Reduce dependencies on external software in the Swiss Post system)

Measure A.10 aims to ensure the external software libraries are only used with "good reason". Swiss Post announced this measure was implemented in version 1.4 of the system. We analysed several parts of the system and comment below on the fulfilment of the measure. Overall, the status seems satisfactory though we have raised a few questions (noted in the following subsections) with Post and are currently awaiting their response. Our rankings when given are based on the MVN repository at the time of writing.

A.1 E-Voting Libraries

Makes use of the following dependencies:

- SLF4J API Module - #1 Logging framework
- BouncyCastle - Standard Cryptographic Library
- MapStruct Core - Annotation processor for generating type-safe bean mappers
- Jakarta - #3 XML bindings
- Guava - #1 Core Utilities
- FindBugs - #1 Defect Detection
- Jackson - #1 JSON library

Summary: The dependencies used are very common and seem well justified.

A.2 Data Integration Service

- Jakarta - #3 XML bindings

Summary: The dependency used is very common and seems well justified.

A.3 Voting Server

- Logback - #2 Logging framework
- Jackson - #1 JSON library
- Caffeine Cache - #1 Caching Implementation

- Guava - #1 Core Utilities
- HikariCP- #1 in JDBC Pools
- Netty - Network programming framework
- Jakarta - #3 XML bindings
- ActiveMq - Common messaging framework
- BouncyCastle - Standard Cryptographic Library
- Quartz Scheduler - #1 Job Scheduling
- SLF4J API Module - #1 Logging framework
- Springframework - Common Framework

Questions:

- Is it necessary to have both SLF4J API Module and Logback?

A.4 Secure Data Manager

- Logback - #2 Logging framework
- Jackson - #1 JSON library
- Guava - #1 Core Utilities
- OrientDB - Database management system
- Netty - Network programming framework
- Jakarta - #3 XML bindings
- Zip4j - Common compression library
- Tomcat - Common webserver
- BouncyCastle - Standard Cryptographic Library
- Jakarta - #3 XML bindings
- SLF4J API Module - #1 Logging framework
- Springframework - Common Framework

A.5 Voting Client and Portal

- jquery - common javascript library

A.6 Control Components

- Jackson - #1 JSON library
- Guava - #1 Core Utilities
- HikariCP- #1 in JDBC Pools
- Jakarta - #3 XML bindings
- BouncyCastle - Standard Cryptographic Library
- SLF4J API Module - #1 Logging framework
- Springframework - Common Framework

A.7 Verifier

- Jackson - #1 JSON library
- Jsurfer - Json Path processor
- Guava - #1 Core Utilities
- Commons-codec - #1 in Base64 Libraries
- Jakarta - #3 XML bindings
- Zip4j - Common compression library
- Tomcat - Common webserver
- MapStruct Core - Annotation processor for generating type-safe bean mappers
- SLF4J API Module - #1 Logging framework
- Springframework - Common Framework

Questions:

- Jsurfer seems less common than other dependencies used, is it necessary?

B Analysis of A.20 (In a public examination, ballots can be set up based on the eCH files)

Measure A.20 calls for

> In a public examination, ballots can be set up based on the eCH files.

While we have not tested a large variety of custom eCH files, we believe the tooling between the `data-integration-service` and `evoting-e2e-dev` mean this requirement is adequately supported.

Running a test ballot remains a somewhat finicky task due to the number of tasks in the various scripts which depend on external servers to respond as well as dependences on the configuration of the local machine. There is certainly room for improvement in making the various scripts less brittle but current status is within the norm for scripts of similar complexity.

C Analysis of Scope 2 Requirements

In this appendix we will go over each requirement within parts of d, e, and f of Scope 2 either providing a summary of the status, or for some requirements in e and f giving a reason for not examining it.

C.1 d) Assess the alignment between software development products

The alignment between the products is mostly acceptable. The main issue is that the specification of the protocol is missing details (see our remarks in the summary of this report) which are security critical and which the implementation does include.

C.1.1 24.1.9, Traceability between functional specifications and security requirements is guaranteed to interface level.

The implementation does ensure various security requirements at the interface level though validating requests before they are processed. However, details of these validations are mostly not included in the specification.

C.1.2 24.1.11, Traceability between the entire source code and the specifications of the security functions is ensured and their correspondence is evident

Related to the previous requirements, it is easy to follow the specification to the code but the other direction is not traceable due to missing details in the specification.

C.1.3 25.1.3, A description must be provided of the link between the legal requirements and the cryptographic protocol, the specifications and the documentation of the architecture.

This link is provided in the various specifications.

C.1.4 25.2.8 The cryptographic protocol, specification, design and source code are aligned.

This does not hold for the reasons described above, namely missing details in the specification of the protocol.

C.2 e) Assess the implementation of the protocol

C.2.1 2.5, Individual verifiability

We are not aware of any unpatched vulnerabilities affecting individual verifiability, there is consequently no known reason why this requirement is not satisfied.

The most recent (now patched) vulnerability, we are aware of, affecting individual verifiability was the issue resulting from inadequate signature verification of the election public key which was addressed in version 1.0.0 of the system. There have been several vulnerabilities discovered since but none which were confirmed to affect individual verifiability; for example, `https://gitlab.com/swisspost-evoting/e-voting/e-voting/-/issues/4` and the issue raised in Appendix B of our report of November 2022 [2].

C.2.2 2.6, Universal verifiability

We are not aware of any unpatched vulnerabilities affecting universal verifiability, there is consequently no known reason why this requirement is not satisfied.

The most recent vulnerability, which we believe affected universal verifiability, is presented in Section 4.2 of this report. This issue was patched in version 1.4.3 of the system.

C.2.3 Privacy

We are not aware of any unpatched vulnerabilities affecting privacy, there is consequently no known reason why this requirement is not satisfied.

The most recent (now patched) vulnerability, we are aware of, affecting privacy appeared in version 1.4.0 of the system, see Sec. 4.1.1 of this report for details.

C.2.4 2.12.1, Only one vote can be cast with the authentication credentials assigned to a voter.

Within the threat model of universal verifiability this requirement is enforced by various checks within the verifier which enforce that at most one ballot is tallied for each credential. During vote casting this is enforced by the honest control component only accepting at most one vote per credential.

C.2.5 2.12.2, The person voting enters their vote on the user device

This occurs on the page of the voting client which is called, in German, Stimme erfassen.

C.2.6 2.12.3, The person voting can change the vote up to the point of confirming the decision to cast it and can check the vote against a summary.

This occurs on the page of the voting client which is called, in German, Kontrollieren and verschlüsseln.

C.2.7 2.12.4, After the person voting has had the opportunity to check the vote against the summary, he or she confirms on the user device that he or she wants to cast the vote as entered

This occurs on the page of the voting client which is called, in German, Kontrollieren and verschluüsseln.

C.2.8 2.12.5, The proofs of correct voting under Number 2.5 must be divided into at least two sequential items of proof. Any indication presented as an item of proof must make a genuine contribution to the soundness of the proof referred to in Number 2.5.

The two sequential proofs are the choice return codes and vote cast return code which are separated by the entering of the ballot casting key.

C.2.9 2.12.6, The user device displays the first item of proof to the person voting after he or she has confirmed on the user device that he or she wants to cast the vote.

This occurs on the page of the voting client which is called, in German, Verifizieren and abgeben.

C.2.10 2.12.7, The user device will not display the next item of proof to the voting person until the voting person has entered into the user device that the previous item of proof is correct.

This is prevented by the control components not releasing the vote cast return code until they have confirmed the ballot casting key.

C.2.11 2.12.8, By confirming that the penultimate item of proof is correct, the voting person confirms his or her decision to cast the vote definitively.

This is realised within the system by the voter entering the ballot casting key.

C.2.12 2.12.9, The group of control components registers the vote as having been cast in conformity with the system when it has received confirmation of the definitive decision to cast the vote.

This is realised by the voter id being added to the list of confirmed votes after the control components process the confirmation key.

C.2.13 2.12.10, When the person voting has checked the last item of proof as being correct, the voting process is complete. The last item of proof should be made particularly easy to check, by limiting the check as far as possible to the correct display of a single code or other simple indication.

The last item of proof is the Vote Cast Return Code which is a single code of 8 digits.

C.2.14 2.12.11, If voting data are imported, a setup component or a print component must no longer be considered trustworthy from that point on.

From the explanatory report it is clear that this requirement is meant to restrict the usage of the setup component and print component in the voting and tally phase. The compliance with this requirement is complicated by the fact that the software product, namely the Secure Data Manager (SDM), is used both as the setup component and in the tally phase; however, the architecture document makes clear that the setup and tally instances of the SDM are separate. We believe this requirement is satisfied.

C.2.15 2.13.1, Wherever possible, building-blocks are used that are in widespread use worldwide and have been thoroughly scrutinised by experts. Standards, reference projects and academic publications can be used as a benchmark. Derogations and cases of doubt must be dealt with separately in the context of the risk assessment referred to in Article 4.

The cryptographic building-blocks used in the system (encryption schemes, hash functions, zero-knowledge proofs, etc) are among the most common variants. There has been some discussion among the reviewers about the prime testing and random number generation which relate to this requirement, however even in these cases the options Post has used are in relatively common use.

C.2.16 2.13.2, 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 believe this requirement is not met for the reasons discussed in the summary of this report.

C.2.17 3.17, Trustworthy components may perform only the intended operations.

This requirement relates to both restrictions the (computer) system enforces and to restrictions imposed by the human operators; we will comment only the restrictions imposed by the computer system. The requirement is also complicated by the fact that many of the components are considered trustworthy in at least one threat model. We list the components which are considered trustworthy in at least one model below:

- Set-up component
- Print component
- Control component
- Auditor's technical aid
- User device (voting client)

We will not attempt to analyse the print component which is not part of the system as released by Post. The control components operate without much human interaction and we have already discussed restrictions on their execution in Sections 1.4 and 4.1. The verifier (auditor's technical aid) likewise has very limited human interaction, we are not concerned that the information available to it should allow any significant unintended operations. This leaves both the setup-component and voting client as the components which are most significant for this requirement.

Set-up Component The set-up component is a specific instance of the `Secure Data Manager`; we analysed the frontend and backed of this component to see what operations it allows in set-up component mode. The `Secure Data Manager` consists of 19 micro-services allowing specific operations which we go through below commenting on the availability in set-up mode and if this is an intended operation in this mode.

To summarise the list below, in general the restrictions are appropriate however four online services are not restricted from usage during set-up

mode. We do not consider this to be significant because the air-gapped set-up component could not physical connect online; moreover, the front-end of the SDM provides guidance to avoid unintended operations.

Online - ComputeController This controller is only available in online mode. It asks the online control components to compute the encrypted long Return Code Shares.

Online - DownloadController This controller is available in all modes of operation.

Online - MixDownloadController This controller is available in all modes of operation. Downloads the ballot boxes and associated data created by the control components.

Online - RequestCcKeysController This controller is only available in online mode. It asks the online control components to provide key shares.

Online - UploadController This controller is available in all modes of operation. Uploads the ballots and voting cards sets to the voting server.

Online - VotingServcerHealthController This controller is available in all modes of operation. It provides updates on the health of the voting server.

Setup - CollectVerifierDataController This component is available only in set-up mode. It prepares the data for the verifier.

Setup - ConstituteElectoralBoardController This component is available only in set-up mode. Takes the electoral board passwords and produces the election keys.

Setup - GenerateController This component is available only in set-up mode. Generates ballot data.

Setup - GeneratePrintFileController This component is available only in set-up mode. It prepares the data for the print office.

Setup - PreConfigureControllers This component is available only in set-up mode. Generates the encryption group, small primes, prime mapping tables and setup keys.

Setup - PreComputeController This component is available only in set-up mode. It generates the verification data.

Shared - DataExchangeController This controller is available in all modes of operation. Imports and exports data between the SDM instances in the form of zip files.

Shared - MetricsController This controller is available in all modes of operation. Returns the processor name.

Shared - SummaryController This controller is available in all modes of operation. Returns a summary of the configuration.

Shared - WorkflowController This controller is available in all modes of operation. Returns data on the current workflow.

Tally - CollectVerifierDataController This controller is only available in tally mode. It prepares the data for the verifier.

Tally - DecryptController This controller is only available in tally mode. It performs the shuffling and decryption.

Tally - ValidateElectoralBoardController This controller is only available in tally mode. It validates the electoral board member passwords.

Voting Client We will defer in an in-depth analysis of this requirement until a future date; we lack sufficient expertise with the frameworks used to analyse the possible control flows in the time available.

C.2.18 25.1.2 All cryptographic protocol requirements across all work products associated with the software development process must be traceable.

As discussed in 2.d the traceability of protocol requirements to the code is acceptable but the requirements are underspecified.

C.3 f) Assess the functionalities

We did not assess the following requirements with part f of Scope 2 for the following reasons:

Reason	Requirements
The requirement relates to usability	4.5, 4.6, 25.7.2, 25.7.3
The requirement relates to operational procedures	3.13, 4.4, 4.9, 4.10, 4.11, 4.12, 5.1, 8.11, 11.1, 11.5

C.3.1 4.1, The person voting must declare that he or she is aware of the rules on electronic voting and of his or her own responsibilities.

This is enforced by the voting client on a page which, in English, translates to Explanation and legal regulation.

C.3.2 4.2, Before casting a vote, the person voting is notified that he or she is taking part in a ballot in the same way as voting by post or voting in person at the ballot box. The person voting may only cast his or her vote after confirming that he or she has taken note of this.

It was not clear to us that this requirement was met.

The text in the voting client does seem to make it clear that this is taking part in a ballot in the same way as voting by post or voting in person at ballot box. However, we could not see any part of the process where the voter specifically confirms they have taken note of this.

– *Status* –

Post has updated the web form with a checkbox specifically requesting confirmation from the voter of this point.

C.3.3 4.3, When voting, the person voting is requested to check the proofs in accordance with Number 2.5 against the verification reference and to report any doubts as to its correctness to the canton.

The honest voting client does do this and we assume this is also true of the voting cards the voter receives but we have not verified this.

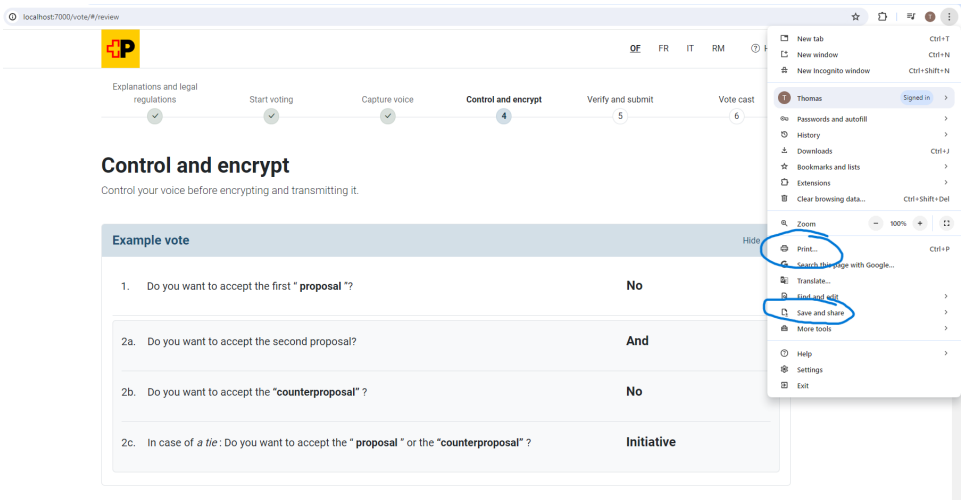

Figure 2: Chrome's print screen and save features

C.4 4.7, The system does not offer the person voting any functionality allowing them to print out or store their vote.

The system as implemented by Post does not have functionality allowing the printing or saving of votes.

However, most web browsers (though which the client will be accessed) have such features; see for example Fig. 2. Moreover, common operating systems include support for screen captures and, more worryingly, screen recording. While the system does not directly offer this functionality, in the environment in which it will be used, these features are easily accessed without any significant expertise or assistance.

C.5 4.8, The person voting is not shown any information after the voting process is completed about the content of the vote that has been encrypted and cast.

Provided cast here is interpreted as confirming the vote, then this requirement is satisfied.

C.6 9, The electronic voting channel is only available during the permitted period.

This is enforced by the control components based on the configuration they received.

C.7 10, Votes not cast in conformity with the system are not stored in the electronic ballot box.

This is ensure by the various validations performed by the control components based on the configuration they received. Voting for a party list without any candidates is currently possible but these ballots will be removed during tally.

C.8 11.6, The system allows the polling card to be used to determine whether someone has cast an electronic vote.

The voter id on the polling card can be used to check the result for the presence of a corresponding vote. The lack of details in Alg. 6.10 of the system specification (RequestProofNonParticipation) which realises this requirement was concerning; it appeared (based on the description of the context) to check for participation only in a single ballot box but it is unclear how it knows which ballot box to check. This may have lead a voter to be told that no ballot was cast on their behalf when in fact one was but in a different ballot box than was checked; we hope that the operational procedure for this check does not exhibit this issue but the specification should, notwithstanding, be updated.

– *Status* –

Post updated the specification in Version 1.4.3 addressing this issue.