

# A Review of the Swiss Post Voting System

Johannes Müller

January 10, 2025

## 1 Overview

**Scope and objective.** In this review, I examine the system specification of the Swiss Post voting system (version 1.4.1.1) to assess the extent to which the system meets the legal security requirements of the regulation. I studied the conceptual design of the system and its cryptographic building blocks. My particular focus was on the setup and voting phases, since these parts of the system are the most specific to the Swiss Post voting system and its election scenario, while the tallying phase is more system-independent and uses more generic techniques.

To complement my view (e.g., to get more information about the underlying cryptographic hardness assumptions or about the specific implementation), I also looked, whenever necessary, at the corresponding documents provided by the Swiss Post (namely *Computational Proof of Complete Verifiability and Privacy*, version 1.3.0, *Infrastructure Whitepaper of the Swiss Post Voting System*, version 2024-03-06, and *Cryptographic Primitives of the Swiss Post Voting System*, version 1.4.1), or at the code as published at <https://gitlab.com/swisspost-evoting>.

**Summary.** At the technical level, the result of my analysis is positive. According to my evaluation, the system as specified in Version 1.4.1.1 fulfills the security properties under the stated assumptions, as claimed by the Swiss Post.

However, there are certain aspects in which the presentation and security of the system could be improved. I will discuss these below.

## 2 Presentational remarks

In this section, I discuss presentation issues in the system specification and make a proposal for improving presentation quality, which could also help simplify security analysis.

**Issue: Inconsistent level of abstraction.** The abstraction level of the system specification is not consistent throughout, but constantly changes between details of the concrete implementation, more general black-box building blocks,

and rather vague descriptions in the body text. To give just one of many examples, in Algorithm 4.6, the black-box building blocks (`GetCiphertextProduct`, `GetMessage`) are mixed with details of the concrete instantiation (group parameters  $p, q, g$ , exponentiated objects, etc.).

At the same time, the system specification does not explicitly state, for example, that the underlying public-key encryption scheme is ElGamal's scheme (even if this becomes clear in the course of reading). In addition, specific operations (such as `HashAndSquare`) are employed whose use is not explained and whose specification is not referenced in the cryptographic primitives document.

For me, the mixture of different levels and the lack of introduction of central operations made it much more difficult to understand and check the system. In particular, I had to constantly switch back and forth between the specification and the document of the cryptographic primitives.

**Issue: Missing description of ideas.** While in some places the ideas behind the building blocks are explained, in some central places there is no explanation of why the building block has been defined as it is. This applies, for example, but not exclusively, to the following parts:

- Encoding of voting options (Section 3.5.2): In particular, a link to and explanation of the SGSP hardness assumption is missing.
- `SetupVoting` (Section 4.1): Fig. 6 und algorithms 4.1, 4.2 (e.g., why do you use `HashAndSquare`?), 4.5 and 4.6 (e.g., why do you use exponentiation?), 4.7 and 4.9 (e.g., why do you use symmetric encryption?).
- `SetupTally` (Section 4.2): Fig. 7.
- `SendVote` (Section 5.2): Fig. 9, Algorithm 5.4 (e.g., why do you use exponentiation?), 5.6 (e.g., why do you use a specific exponentiation in line 7 and not the more generic algorithm?), 5.7, 5.8 (e.g., why do you use `HashAndSquare`?), 5.9 (e.g., why do you use symmetric encryption? and why do you split ciphertexts?).
- ... and similarly for the remaining phases.

Firstly, this leads to a much longer time to understand the specification, and secondly, it opens up the possibility for misinterpretation.

**Issue: Unclear statements proved by zero-knowledge proofs.** In the system specification, the statements proved by the various zero-knowledge proofs are only vaguely described. To give an example, Section 3.8 (Proof of Correct Key Generation) states that

To prevent this attack, each participant creates a Schnorr proof of knowledge of their secret key matching the distributed public key.

However, it is not specified what exactly this means. While Algorithm 3.23 states that "for all algorithms, see the crypto primitives specification", the exact relation to be proven remains unclear. The same is true for the other zero-knowledge proofs.

**Proposal for improvement.** I propose the following fundamental change to the presentation of the specification:

The system is first specified on a level that abstracts from the concrete instantiation. To this end, the various cryptographic components (public-key encryption, zero-knowledge proofs, etc.) are introduced as black boxes with the desired properties. The next step is to show how these black-box primitives are instantiated. For this, one can then go into the specific cryptographic details.

I think that in this way, it will be easier to describe the essential design ideas of the overall system, and thus develop a better understanding of the system and evaluate security at a conceptual level.<sup>1</sup> This change can also be useful to assess whether the abstract symbolic models for the automated verification (which also abstract away from the concrete instantiation!) correctly represent the real voting system or whether there are significant gaps between them that may have been overlooked.

### 3 Technical remarks

As I wrote above, when I studied the document that specifies the voting system, I was able to convince myself that the system meets the desired security properties. However, I would like to discuss two technical aspects in the following.

**Uncommon hardness assumption for vote privacy.** The hardness assumption for vote privacy, the Subgroup Generated by Small Primes (SGSP) problem, is an unusual hardness assumption that has hardly been studied. Therefore, I think it is good that it is planned to replace this assumption with a more established and better understood hardness assumption.

**Limitations of the return code technique.** Since the individual verifiability of the current system is based solely on return codes, there is no guarantee for write-in candidates (or for elections with more complex ballots) that the vote of the voter has been processed by the voting device as intended. I therefore suggest that further verifiability methods (such as using second devices) be enabled and implemented in the future. In particular, this could protect against a corrupted voting device (using suitable manipulation methods) causing a voter to write in the name of the candidate instead of choosing from a list, thus undermining the possibility of individual verifiability.

---

<sup>1</sup>Specifically, although I consider myself an expert in secure e-voting systems, it took me more than seven days to understand the specification in its entirety and to convince myself of its claims. I hope that an appropriate revision will reduce this effort and make it easier for non-experts to understand.