2022 Re-evaluation of the Swiss Post e-Voting System (Addendum)

Audit Scope 1: Cryptographic Protocol

Aleksander Essex

Department of Electrical and Computer Engineering
Western University, Canada
aessex@uwo.ca

November 21st, 2022

Submitted to the Swiss Federal Chancellery

Table of Contents

1 Introduction and Summary of Findings 1
2 Key Recommendations for Unresolved Issues 2
3 Scope of Re-evaluation 4
3.1 Documents Examined 4
4 Response to Swiss Post VEs 8
4.1 Hash Functions 9
4.2 Key Length / Security Levels 10
4.3 Start Voting Key 10
4.4 General Improvements 11
4.5 Primality Testing 17
4.6 Progress on Key Recommendations of 2021 Preliminary Report 19
4.7 Progress on Key Recommendations of 2021 Final Report 21
4.8 Unimplemented Findings 22
4.9 Out of Scope 23
4.10 Length-based Attacks 23
5 Addendum: Remaining VEs Re-examined 24
References 28
A Author Bio 28

Version History

This report supersedes the report entitled "2022 Re-evaluation of the Swiss Post e-Voting System," dated September 30th, 2022. The primary change is the inclusion of an addendum section (Section 5), which re-examines the remaining issues (VEs) outstanding from the September 30th report.

1 Introduction and Summary of Findings

Following an examination in 2021, I was retained by the Swiss Federal Chancellery in 2022 to re-examine the e-voting solution of Swiss Post in relation to the Chancellery's ordinances on electronic voting (OEV) in regard to the cryptographic components and their respective security properties (Scope 1).

My main goal of this re-examination was to assess whether (and to what degree) protocol and documentation changes made by Swiss Post addressed the issues identified in my 2021 Final Report [7]. Swiss Post identified 75 issues in my Final Report (November 2021). Of these, two issues were separated into 4 sub-issues each, for a total of 81 issues.

Swiss Post has already responded to and resolved 8 issues at the time of writing the Final Report (November 2021). Since then, Swiss Post has explicitly responded to 43 issues in the context of this re-examination (September 2022). Subsequently, I have added an addendum (to this document) to re-examine any relevant changes in the documentation to the remaining 30 issues (November 2022). This report covers the re-examination of these outstanding 73 issues.

Issues addressed
Summary: 73 of the outstanding issues identified in my Final Report (November 2021) are re-examined. 43 issues were explicitly responded to by Swiss Post for this re-evaluation. These were re-examined in September 2022. The remaining 30 were re-examined in an addendum included in this version (November 2022).

Resolved
Summary: 59 of 73 issues have been resolved (ca. 80%).

Unresolved
Summary: 14 of 73 issues remain unresolved (ca. 20%).
Minor: 8 issues are classified as minor, requiring small notational changes, additional clarification or analysis.
Medium: 6 issues are classified as medium, requiring better alignment with better/best practices or established security guarantees.
Major: No issues are classified major (leading to an immediate vulnerability).

2 Key Recommendations for Unresolved Issues

The re-evaluation identified several opportunities for improvement. These recommendations are in no particular order and are as follows:

1. **VE-4961**: Recursive Hash – Infinite Input (Section 4.1)
   - **Classification**: Minor.
   - **Recommendation**: Minor notional clarification.
2. **VE-4940**: KDF Parameterization (Section 4.4)
   - **Classification**: Minor.
   - **Recommendation**: Additional analysis is needed to map the memory-hardness of Argon2 to a concrete parameterization at the given security-level (akin to PBKDF2).
3. **VE-5136**: Implications of Quantum Computing (Section 4.9)
   - **Classification**: Minor.
   - **Recommendation**: Acknowledge and discuss the privacy and integrity implications if an at-scale implementation of Shor's algorithm becomes feasible. Note this recommendation differs from Swiss Post's interpretation of my original remarks. To be clear: I do not believe a total redesign of the system with post-quantum cryptographic primitives is necessary or feasible.
4. **VE-5422-1**: Ensuring Function Inputs Have Expected Form (Section 4.6)
   - **Classification**: Medium.
   - **Recommendation**: Make input validation explicit in the primitives and verifier specification documentation. Do not rely on checks in the software implementation to stand as the implicit documentation. Its purpose, criticality, and methodology should also be made explicit in the specification.
5. **VE 5423-1**: Under-specified Voter Authentication (Section 4.7)
   - **Classification**: Medium.
   - **Recommendation**: Provide a list of the necessary and sufficient elements and factors needed to cast a ballot. Clarify if any of these design decisions are downloaded to the user (cantons) and provide some basic guidance on concrete approaches.
6. **VE-4969**: Primality Testing Method (Section 4.5)
   - **Classification**: Medium.
   - **Recommendation**: In the absence of formal security guarantees for the Baillie-PSW primality testing algorithm, use the Miller-Rabin algorithm or combine the two. See the discussion on the implications of primality testing in this context.

7. **VE-5422-3**: Increasing Security Levels (Section 4.7)
   - **Classification**: Medium.
   - **Recommendation**: Discontinue the use of the 112-bit security level, or give the user a specific warning/acknowledgment that ECRYPT [15] and NIST [3,6] do not recommend this level for security lifetimes protection beyond 2030. Work with the Chancellery to define an explicit "security lifetime" for election data.
8. **VE-4971**: Smallest generator (Section 5)
   - **Classification**: Minor.
   - **Recommendation**: Justify or remove the requirement in `GetEncryptionParameters` in the primitives specification to select the smallest generator of $\mathbb{G}_q \subset \mathbb{Z}_n^*$.
9. **VE-5117**: Safeguarding against deployment of testing-only security level (Section 5)
   - **Classification**: Medium.
   - **Recommendation**: Add an explicit check to the verification specification (e.g. in `VerifyEncryptionParameters`) to ensure encryption parameters conform to the bit lengths specified in the *standard* or *extended* security levels.
10. **VE-5123**: Explain mixnet architecture design choice (Section 5)
    - **Classification**: Minor.
    - **Recommendation**: Explain to what degree the choice of a Bayer-Groth mixnet architecture is arbitrary, or is rooted in a specific performance, legal, reputational or similar consideration.
11. **VE-5124**: Unsupported claim about mixnets (Section 5)
    - **Classification**: Minor.
    - **Recommendation**: Add a reference to a literature review showing verifiable mixnets "underpin most modern e-voting schemes," or modify the claim.
12. **VE-5126**: Implied modular reduction in subscript (Section 5)
    - **Classification**: Minor.
    - **Recommendation**: Clarify that subscripts of elements $\pi_{i+offset}$ are intended to be reduced modulo the list size N.
13. **VE-5421**: Voter authentication (Section 5)
    - **Classification**: Medium.
    - **Recommendation**: Additional clarity needed relating to the expected real-world voter authentication credentials that would be necessary and sufficient to cast a ballot and in Swiss Post's opinion, why it satisfies the associated requirements of the OEV.
14. **VE-5428**: Fix quotation marks (Section 5)
    - **Classification**: Minor.
    - **Recommendation**: N/A.

3 Scope of Re-evaluation

Following the Scope-1 audit in 2021,[1] Swiss Post collected and catalogued the respective findings of the reviewers. This included my report [7], the reports of Basin [4], Haenni et al. [10], Ford [8], and Haines, Pereira, and Teague [11]. Swiss Post catalogued findings using the Jira[2] issue tracking software and assigned each finding a 4-digit Jira Ticket number of the form VE-*XXXX*. Swiss Post sent the experts a Jira Ticket Inventory List (see Section 3.1).

Of the 329 tracked issues, 75 directly pertained to my report. Two of these issues I further sub-divided into 4 individual sub-issues each, for a total of 81 issues. Of these, 8 issues were previously resolved at the time of my November 2021 Final Report. The scope of this re-evaluation is limited to re-examining the outstanding 73 issues. This report initially covered the 43 issues directly responded to by Swiss Post in its re-examination documents. An addendum (in this version) re-examines the remaining 30 issues.

3.1 Documents Examined

During this engagement, I have examined Swiss Post documents at three separate snapshots in time: my Preliminary Report (September 2021), my Final Report (December 2021), and this Re-examination Report (September 2022). The following information identifies which version of Swiss Post's documentation was examined at which point in time. For each document, this includes:

- A description of the given document.
- For each of my reports, the version and date of publication of the Swiss Post document examined in that report.
- A URL to the git-controlled document. If the document is not publicly available, the filename is provided for reference.

Primitives Specification

Description: Pseudocode specifications of cryptographic functions used by the Swiss Post system. Referred to throughout this document as the *primitives specification*.

Report	Version Examined	Date Published
2021 Preliminary Report	0.9.5	2021-06-22
2021 Final Report	0.9.8	2021-10-15
2022 Re-Examination	1.0.0	2022-06-24

Available: https://gitlab.com/swisspost-evoting/crypto-primitives/crypto-primitives/-/blob/master/Crypto-Primitives-Specification.pdf

[1] https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/tree/master/Reports/Examination2021

[2] https://www.atlassian.com/software/jira

System Specification

Description: Document describing the steps, phases and procedures of setting up, executing and verifying an election using the Swiss Post system. Referred to in this document as the *system specification*.

Report	Version Examined	Date Published
2021 Preliminary Report	0.9.6	2021-06-25
2021 Final Report	0.9.7	2021-10-15
2022 Re-Examination	1.0.0	2022-06-24

Available: https://gitlab.com/swisspost-evoting/documentation/-/blob/master/System/System_Specification.pdf

Protocol Specification

Description: Document describing the cryptographic primitives and protocols of the Swiss Post system and the various security proofs thereof. Referred to in this document as the *protocol specification*.

Report	Version Examined	Date Published
2021 Preliminary Report	0.9.10	2021-06-25
2021 Final Report	0.9.11	2021-10-15
2022 Re-Examination	1.0.0	2022-07-29

Available: https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/Protocol/Swiss_Post_Voting_Protocol_Computational_proof.pdf

Verifier Specification

Description: Document describing the algorithms and checks that an independent verifier of the Swiss Post system would undertake to verify the correctness and validity of the election parameters, proofs, and ultimately, the election results. Referred to in this document as the *verifier specification*.

Report	Version Examined	Date Published
2021 Preliminary Report	Not available	N/A
2021 Final Report	0.9.1	2021-10-15
2022 Re-Examination	1.0.1	2022-08-19

Available: https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/System/Verifier_Specification.pdf

Jira Ticket Inventory List

Description: Document presenting each original finding of the experts from 2021 as a sequence of Jira ticket numbers (RE-XXXX) with a screenshot of the relevant text of the expert's report.

Filename: `JIRA Exports-v17-20220127_164723.docx`

Available: Not available online. Sent to examiners privately by Swiss Post WebTransfer. Exported January 27, 2022. Received March 31, 2022.

Part I Re-Examination Documents

Description: Documents describing Swiss Post's progress responding to issues identified in 2021. Consists mainly of a list of Jira ticket numbers that have been addressed. For each ticket number, a title identifies the relevant document and section of the change. In some instances, a description of the change is given.

Report	Version Examined	Date Published
2022 Re-Examination	X04.00	2022-04-14
2022 Re-Examination	V01.00	2022-06-24

Filenames:

- `ReExamination_Part1.pdf` (X04.00)
- `ReExamination_Part2_24.06.2022.pdf` (V01.00)

Available: Not available online. Sent to examiners privately by email.

Response to Examination Report by Aleksander Essex – Scope 1 Cryptographic Protocol

Description: Response by Swiss Post to a subset of findings made in my 2021 Final Report.

Report	Version Examined	Date Published
2022 Re-Examination	N/A	2022-04-20

Available: https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/Reports/Examination2021/Scope-1-Essex-Examination-eVoting-System-Response-Swiss-Post.pdf

Federal Chancellery Ordinance on Electronic Voting (OEV)

Description: Primary legal document outlining relevant technical requirements, particularly the cryptographic requirements for complete verifiability.

Report	Version Examined	Date Published
2021 Final Report	Draft	2021-04-28
2022 Re-Examination	N/A	2022-05-22

Available: https://www.fedlex.admin.ch/eli/cc/2022/336/en

4 Response to Swiss Post VEs

I identified the following VEs associated with my 2021 report(s) based on Swiss Post's Jira Ticket Inventory List.

The scope of re-examination in this report corresponds to those VEs directly acknowledged in Swiss Post's re-examination documents. Because the complete mapping of VEs to the original text of my report(s) is in a non-public Jira Ticket Inventory List, this section is intended mainly for Swiss Post's reference.

VEs Previously Resolved in my Final Report (Nov. 2021)
4925, 4926, 4927, 4938, 4959, 4965, 5122, 5127

VEs Covered in my Initial Re-examination Report (Sept. 2022)
4924, 4938, 4940, 4941, 4942, 4946, 4947, 4948, 4949, 4950, 4951, 4952, 4953, 4954, 4955, 4956, 4957, 4958, 4960, 4961, 4962, 4964, 4966, 4968, 4967, 4969, 4970, 5115, 5116, 5120, 5121, 5425, 5128, 5130, 5136, 5422-1, 5422-2, 5422-3, 5422-4, 5423-1, 5422-2, 5422-3, 5422-4

Additional VEs re-examined in this Addendum Report (Nov. 2022)
4928, 4963, 4971, 5117, 5118, 5119, 5123, 5124, 5126, 5129, 5131, 5132, 5133, 5134, 5135, 5421, 5424, 5428, 5429, 5430, 5431, 5432, 5433, 5434, 5435, 5436, 5437, 5438, 5439, 5440

Re-examination Structure. The VEs re-examined for this re-evaluation are presented in the following sub-sections. Each re-examined VE consists of the following elements:

- A title identifying the VE and a status indicating whether (in my judgement) the item has been resolved (color-coded green), or the item remains unresolved (color-coded gray).
- A summary of the issue as it was identified in my 2021 report, with a back reference to the relevant page number.
- The action taken by Swiss Post to address the issue.
- In some cases, a remark is included with my response to Swiss Post's action taken.

Some issues had already been resolved by Swiss Post as of my November 2021 Final Report. Those issues had not been explicitly mapped to VEs at that time. As part of this re-examination, I have identified and listed previously resolved VEs in the table above.

4.1 Hash Functions

This section pertains to changes to Swiss Post's recursive hash function as defined in the primitives specification. See: Swiss Post Re-Examination note, Section 2.2, X04.00, April 2022.

VE-4961: Recursive Hash – Infinite Input (Unresolved)
Summary: Unusual wording about avoiding "infinite input in practice" (pg. 14 [7]).
Action Taken: Unchanged.

VE-4962: Recursive Hash – Pseudo-random Function (Resolved)
Summary: Define "pseudo-random hash function" (pg. 14 [7]).
Action Taken: Wording in the primitives specification changed to "cryptographic hash function."

VE-4964: Recursive Hash – Domain Separation (Resolved)
Summary: Domain separation between decimal and byte representations (pg. 15 [7]).
Action Taken: N/A. Previously resolved in 2021.

4.2 Key Length / Security Levels

This section pertains to changes to Swiss Post's use of key lengths and security levels in the primitives specification and start voting key (SVK) in the system specification. See: Swiss Post Re-Examination note, Section 2.4, X04.00, April 2022.

The re-evaluation note ascribes this issue to VE-4970, however, I believe VE-4968 was the intended number. Both VEs are included below for completeness.

VE-4968: Encryption – Security Levels (Resolved)
Summary: Recommended use of the term "security level" instead of "strength" (pg. 15 [7]).
Action Taken: A table was added to the primitives specification making this explicit (Table 2).

VE-4970: Symbol Overloading in `GetEncryptionParameters` (Resolved)
Summary: Overloaded use of symbol λ as security level and `isProbablePrime` testing rounds (pg. 15 [7]).
Action Taken: Algorithm 7.1 of crypto primitives now makes rounds r explicit and distinct.

4.3 Start Voting Key

This section pertains to Start Voting Key (SVK) in the system specification. See: Swiss Post Re-Examination note, Section 2.4.1, X04.00, April 2022.

Swiss Post states it increased the SVK to "128 bits and a length of 24 digits" in the system specification (Sec 3.5, page 25). The approach draws 24 uniform characters from the $\mathbb{A}_{32}$ alphabet to achieve 120 bits of entropy. The Argon2 key-derivation function is used to extend (stretch) this to 128-bits. Table 11 of the system specification does not specify the number of iterations necessary to stretch a 120-bit key to the 128-bit security level.

Recommendation: Make Table 11 explicit by stating how many iterations are required to reach the intended 128-bit security level. This value in PBKDF2 terms is rather straightforward: 256. However, Swiss Post changed KDFs to Argon2, and should provide concrete guidance for parameterizations for the given security levels. Also clarify the term "digits" in this context refers to elements of $\mathbb{A}_{32}$ (i.e., not the decimal $\mathbb{A}_{10}$).

4.4 General Improvements

This section pertains to general fixes and improvements to the primitives specification. See: Swiss Post Re-Examination note, Section 2.7, X04.00, April 2022.

VE-4924: Redundancy Across Documents (Resolved)
Summary: Preliminaries in the system and primitives specifications are largely overlapping (pg. 11 [7]).
Action Taken: Merged into the primitives specification with reference in the system specification.

VE-4938: Missing Modular Reduction (Resolved)
Summary: Algorithm 8.1 (`HashAndSqaure`) in the system specification (Version 0.9.6) does not perform a modular reduction, which could lead to outputs larger than q (pg. 11 [7]).
Action Taken: N/A. Previously resolved in 2021. Moved to primitives specification (Algorithm 4.1, version 1.0.0) and modular reduction included (line 2).

VE-4940: KDF Parameterization (Unresolved)
Summary: Justify choice of 32,000 hash iterations in PBKDF2 (Algorithm 8.2 in version 0.9.5 and Algorithm 8.8 in 0.9.7) in the system specification (pg. 11 [7]).
Action Taken: PBKDF2 was switched to the memory-hard Argon2 KDF (Algorithm 4.5) defined in the primitives specification, however parameters (such as iteration count) are still left mostly abstract with the incomplete suggestion that they "should be chosen taking in consideration the entropy of the input."
Remark: Determining optimal parameterizations Argon2 is not as straightforward as for PBKDF2. Please provide concrete parameterizations to match the *default* and *extended* security levels. The Security Level (Table 2, primitives specification) also lists SHA-256 as the KDF's hash function. However, Argon2 is typically described in the context of the Blake2b hash function and mixing and matching hashes and KDFs in this manner could impact the overall memory-hardness.[3]

[3] https://crypto.stackexchange.com/questions/53260/argon2-with-sha-256-instead-of-blake2

VE-4941: Various Symbol Definition Issues (Resolved)
Summary: Corrections/clarifications needed in several places in the list of symbols in the primitives specification (pg. 12 [7])
Action Taken: Issues with definitions of $\mathbb{G}_q$, $\mathbb{N}^+$, the primality of p, q and the preferred use of the cursive ℓ have all been addressed.

VE-4942: AES-GCM (Resolved)
Summary: AES-GCM initialized at the 128-bit security level with a 96-bit IV (pg. 12 [7]).
Action Taken: Citations were provided in Section 5 of the primitives specification, explicitly justifying this design decision.

VE-4946: Basic Data Types – Alphabets (Resolved)
Summary: Several algorithms conflate the notation of a character c drawn from an alphabet, i.e., $c \in \mathbb{A}_x$, with a string s drawn from a set of strings created from that alphabet i.e., $s \in (\mathbb{A}_x)^*$ (pg. 12 [7])
Action Taken: Kleene stars were added at the appropriate locations in the relevant algorithms.

VE-4948: Verum/Falsum Symbols (Resolved)
Summary: Define *verum* $\top$ and *falsum* $\bot$ truth symbols (pg. 13 [7])
Action Taken: Now explicitly defined in the primitives specification symbol list (pg. 5 in version 1.0.0).
Remark: The statement "Truth value true or successful termination" should use a different font to distinguish *true* (resp. *false*) as a value, e.g., "Truth value `True` or successful termination."

VE-4949: Return / Output Types (Resolved)
Summary: Clarify difference between a function return and a function output (pg. 13 [7]).
Action Taken: Distinction is now defined in Section 1.1 of the primitives specification (pg. 7 in version 1.0.0).

VE-4950: Falsum in Output Sets (Resolved)
Summary: The falsum symbol should be explicitly included in the output set (co-domain) of functions that have a failure mode (pg. 13 [7])
Action Taken: Symbol is now included in output sets of the relevant algorithms in the primitives specification.

VE-4951: Number Formatting – Thousands Separator (Resolved)
Summary: Clarify the apostrophe thousands separator (e.g., 1‘000‘000) for unfamiliar audiences (pg. 13 [7]).
Action Taken: Thousands separator switched to spaces (e.g., 1 000 000) with an explicit acknowledgment (Table 4 in version 1.0.0).

VE-4952: Range Notation (Resolved)
Summary: Define the inclusive/exclusive bounds of the $[0, n)$ range notation (pg. 13 [7])
Action Taken: Addressed previously in 2021.

VE-4953: Byte Length Notation (Resolved)
Summary: `IntegerToByteArray` of primitives specification (Algorithm 2.8, version 0.9.5) calculates byte length as $\lceil \frac{\lfloor x \rfloor}{8} \rceil$, whereas Algorithm 3.1 invokes the redundant `byteLength()` functionality (pg. 13 [7]).
Action Taken: `IntegerToByteArray` updated to use `ByteLength()`, which internally uses the floored quotient notation.

VE-4954: Uniformity of Output Distributions (Resolved)
Summary: The input and output domains, as well as the statistical properties of `randomBytes`, should be stated (pg.13 [7]).
Action Taken: This has been clarified in Footnote 1 in Section 4.1 of the primitives specification (pg. 17, version 1.0.0).
Remark: Function names `RandomBytes`, `ByteLength()` are now also using uppercase naming convention.

VE-4955: Consistent Capitalization of Algorithms (Resolved)
Summary: Capitalization of functions `byteLength` and `randomBytes` is inconsistent with the naming convention (pg. 13 [7]).
Action Taken: `ByteLength` (Algorithm 3.10) and `RandomBytes` (Sec. 4.1) now styled in upper-case, which is consistent with the other algorithms.

VE-4956: `ByteArrayToInteger` Naming (Resolved)
Summary: ToInteger in Algorithm 3.1 (version 0.9.5) should be ByteArrayToInteger (pg. 13 [7]).
Action Taken: Corrected in Algorithm 4.1 (version 1.0.0).

VE-4957: Upperbound vs. Upper bound (Resolved)
Summary: "Upperbound" and "upper bound" variously be used (pg. 13 [7]).
Action Taken: "Upper bound" is now consistently used (e.g., Algorithms 4.1, 4.1 in version 1.0.0).

VE-4958: Integer vs. Number (Resolved)
Summary: Algorithm 3.1 (in version 0.9.5) should more precisely output a random *integer*, not just "number" (pg. 13 [7]).
Action Taken: Addressed in Algorithm 4.1. (in version 1.0.0).

VE-4960: Truncate Function (Resolved)
Summary: Define Truncate function in Algorithms 3.3-5 (version 0.9.5) of primitives specification (pg. 13 [7]).
Action Taken: `Truncate` now defined in Algorithm 4.6 (version 1.0.0).

VE-4966: Primality of p, q (Resolved)
Summary: The primality of p, q (Section 4.1, version 0.9.5) of primitives specification was never stated (pg. 14 [7]).
Action Taken: Corrected in Section 7.1 (version 1.0.0).

VE-4967: Default ElGamal Generator (Resolved)
Summary: Clarification is needed on the origin of the specification of the smallest element of $\mathbb{G}_q$ being used as a generator in the ElGamal cryptosystem (pg. 14 [7]).
Action Taken: Section 7.1 (version 1.0.0) now clarifies "we" (i.e., Swiss Post, as opposed to Tahir Elgamal) require the smallest element of $\mathbb{G}_q$ to be used as the generator.
Remark: There appears to be no particular issue with using small integers as generators.[4] Swiss Post also evidently addresses my remark (pg. 15 [7]) that technically there is no "smallest" group element. Section 7.1 of the primitives specification now correctly defines it as the "smallest *integer* $x > 1$ s.t. x is a generator of $\mathbb{G}_q$."

VE-5120: Exponentiation (Resolved)
Summary: Multi-ciphertext exponentiation in caption of Algorithm 4.6 (version 0.9.5) of the primitives specification makes it sound like single exponentiation (pg. 19 [7]).
Action Taken: Algorithm 7.6 (version 1.0.0) now clarifies: "Exponentiate each ciphertext element by an exponent a."

VE-5121: Define $\bar{*}$ Notation (Resolved)
Summary: Define the $\bar{*}$ notation in `GenVerifiableDecryptions` in Algorithms 4.11 and 6.3 (version 0.9.5) of primitives specification (pg. 19 [7]).
Action Taken: Notation removed in Algorithms 7.11 and 7.12 (version 1.0.0).
Remark: Appears to have been a typo.

VE-5128: Confusion in ϕ Function (Resolved)
Summary: The ϕ symbol is variously used as a function and a value in Section 6 of the primitives specification (pg. 21 [7]).
Action Taken: Relevant variables retain the ϕ symbol. In instances where it is used in the context of a function, the document calls it a "phi-function." See, e.g., Section 9.2 (version 1.0.0) of the primitives specification.

[4] Pierrick Gaudry. About the SGSP problem. Whitepaper, 2022. Available: https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/Protocol/sgsp.pdf

VE-5130: Verify Hash Should be e' (Resolved)
Summary: Algorithm 6.9 Line 6 (version 0.9.5) of the primitives specification: e, the asserted hash of the prover is conflated with e', the reconstructed hash of the verifier (pg. 21 [7]).
Action Taken: Fixed in Algorithm 9.12 (version 1.0.0)

VE-5115: Bounds on Voting Options Prime Products (Resolved)
Summary: The `EncodeVotingOptions` and `DecodeVotingOptions` algorithms do not enforce upper bounds on the product of primes (pg. 11 [7]).
Action Taken: Swiss Post clarifies the control components enforce this.

4.5 Primality Testing

This section pertains to changes to Swiss Post's primality testing in the primitives specification. See: Swiss Post Re-Examination note, Section 2.8, X04.00, April 2022.

VE-4969: Primality Testing Method (Unresolved)
Summary: Specify primality testing method used by IsProbablePrime in the primitives specification (pg. 15 [7]).
Action Taken: Section 4.6 of primitives specification (version 1.0.0) introduces a number of new algorithms (4.15-4.21) specifying how primality testing is to be performed (good). However, they chose to use the Baillie-PSW test [2,13] instead of the standard Miller-Rabin test [12,14]. The motivation extends from Albrecht et al. [1], who demonstrated the ability to compute Miller-Rabin pseudoprimes (composite numbers that improperly pass the primality test) under specific conditions. However, unlike Miller-Rabin, Baillie-PSW has no bounds on the density of pseudoprimes. It's not presently known.
Remark: In the absence of known/proven bounds for (or even the existence of) BPSW pseudoprimes, and given how Albrecht et al. exploited the affected libraries combined with the reversed adversarial setting of this application, Miller-Rabin should still be used, either alone or in combination with BPSW. Additional explanation is given below.

Why Miller-Rabin Should Still be Used for Primality Testing.

It is important to understand the exact mechanism that allowed Albrecht et al. [1] to reliably construct Miller-Rabin pseudoprimes. The issue wasn't so much about prime generation under "adversarial conditions" as it was about improper shortcuts being taken in the code of the affected libraries.

The probability that a random number n is a strong probable prime, i.e., passes t iterations of the Miller-Rabin test, is $< (1/4)^t$, however each of the t tests requires a random base in the range $2 \leq b_i < (n-1)$. The flaw Albrecht et al. exploited was an adversary's ability to *predict* which bases a verifier would choose in their Miller-Rabin test. This was possible because the affected libraries improperly used fixed (static) bases or randomly sampled from a small set of predictable bases (Apple, LibTomMAth, PyCrypto), allowing Albrecht et al. to compute integers that were strong pseduoprimes at each of those bases.

In the case of GNU GMP 6.1.2, the PRNG used to generate bases was initialized "to a static state[5] and consequently, the bases used in its Miller-Rabin test depend only on n, the number being tested."

The authors conclude by recommending the Baillie-PSW be used as a replacement for Miller-Rabin because no BPSW-pseudoprimes have been discovered to date. Indeed, finding even *one* BPSW pseudoprime remains an open problem in number theory. However, as Swiss Post themselves acknowledge in the primitives specification (Section 4.6, version 1.0.0), no proven bounds exist on the number of BPSW-pseudoprimes below a

[5] https://gmplib.org/repo/gmp-6.1/file/tip/mpz/millerrabin.c

bound $b > 0$, although it has been computationally shown no BPSW pseudoprimes exist below 2^{64}.[6]

A BPSW pseudoprime is both a Lucas pseudoprime and a strong base-2 pseudoprime (i.e., one passing the Miller-Rabin test with a base of $a = 2$). The pseudoprimes produced by these two tests appear to be fundamentally non-overlapping, but the exact reason is not understood. Although BPSW has pseudoprimes below 2^{64}, Swiss Post is generating primes up to 2^{3072}, and the mere "absence of known pseudoprimes" is not a guarantee.

Miller-Rabin, by contrast, has strong bounds *if* sufficiently many random bases are used and are not guessable to an adversary *a priori*. Is this realistically enforceable in software? I would argue yes: The adversarial setting, in this case, is reversed. Swiss Post is the "adversary," and the `isProbablePrime` prime function is only being called in the context of verifying encryption parameters within their own product.

Barriers to exploiting Miller-Rabin. Exploiting the attack of Albrecht et al. [1] is not feasible for several reasons. It would require Swiss Post to be able to reliably predict in advance which bases would be chosen by the user to test a prime candidate, which would only be feasible if Swiss Post *themselves* specified a vulnerable base-selection algorithm—after already drawing everyone's attention to it in the primitives specification (Section 4.6, version 1.0.0). Even if nobody noticed, `GetEncryptionParameters` is structured so that q and $p = 2q + 1$ must be chosen to both be prime, which Albrecht et al. acknowledge "cannot produce malicious parameters in this case... with our current techniques." Furthermore, the attack of Albrecht et al. was not constrained by the requirement that the seed used to generate p, q be semantically meaningful based on the election's name.

Finally, any success at producing a composite p or q would only escape detection within *that* specific implementation since the other vulnerable implementations studied by Albrecht et al. have since been patched.

To summarize why Miller-Rabin should not be abandoned for BPSW:

- The density/existence of Baillie-PSW pseudoprimes have no proven bounds, and finding a counter-example is not currently tied to any hardness assumption. By contrast, Miller-Rabin has well-established bounds for randomly chosen bases.
- The attack of Albrecht et al. [1] on Miller-Rabin requires the adversary to be able to guess the "random" bases ahead of time. The attack fails if bases are properly chosen: (a) from the unrestricted set of possibilities, and (b) independently of the tested n number being tested.
- The adversarial setting of Albrecht et al. [1] attacks specific vulnerable implementations in a general-use setting, whereas Swiss Post themselves are the "adversary," and has a specific constrained use-case, which makes exploitation infeasible for the reasons given above.

[6] Jeff Gilchrist. Pseudoprime Enumeration with Probabilistic Primality Tests, 2013. Available: http://gilchrist.great-site.net/jeff/factoring/pseudoprimes.html

Recommendation:

1. Use Miller-Rabin instead of Baillie-PSW. To avoid the attacks of Albrecht et al., the primitives specification would require `isProbablePrime` to select bases chosen by a non-deterministic, independent seed drawn uniformly from the *full* interval of valid bases (i.e., $2 \leq b_i < (n-1)$).

2. Use a hybrid approach similar to the current GNU GMP implementation: Begin with a BPSW test and finish with a Miller-Rabin test (on a reduced number of rounds). For example, Version 6.2.1 of GMP[7] replaces the first 24 Miller-Rabin tests with a Baillie-PSW test.

4.6 Progress on Key Recommendations of 2021 Preliminary Report

This section pertains to progress on the key findings made in my preliminary report. See: Swiss Post Re-Examination note, VE-5422, Section 2.7, X04.00, April 2022. Of these key findings (pg. 2 [7]), Swiss Post states in response to VE-5422 that "The recommendations have by and large been implemented in the latest primitives spec" (version 1.0.0). However, of those 4 findings, two remain unresolved.

VE-5422-1: Ensuring Function Inputs Have Expected Form (Unresolved)
Summary: Make the primitives specification explicit that all functions must ensure inputs have the expected form, e.g., belong to the expected numeric range, algebraic group, etc. For example, an algorithm accepting an untrusted input $x \in \mathbb{G}_q$ must run an explicit test to ensure $x \in \mathbb{G}_q$.
Action Taken: No change in the primitives specification. Haenni et al. [10] also noted a lack of explicit input validation happening in the code (VE-5514). The code was refactored to perform explicit input validation. It does now appear Swiss Post is doing the necessary input validation in software.
Remark: Input checking/validation is mission-critical. It seems problematic to implement these checks in software if they are not also explicitly reflected in the specification documents. As I have underscored in each of my reports, there is too much historical precedent for implementation errors and misunderstandings to leave it unwritten in the specification. Future Swiss Post developers should be able to re-create a new, correct software implementation from the specification and should not be expected to know how (or even *that*) such checks need to be done. For example, checking $x \in \mathbb{G}_q$ would normally involve checking: (1) $0 \leq x < p$, and (2) $x^q \equiv 1 \bmod p$. However, since p is safe-prime and $\mathbb{G}_q \subset \mathbb{Z}_p^*$ is the subgroup of quadratic residues, the relatively expensive modular exponentiation can be efficiently and effectively substituted by checking the Legendre symbol $(\frac{x}{p}) = 1$. Checks like these should be spelled out in the primitives specification and not left to source code to form the historical record of *what* and *why*.

[7] https://github.com/alisw/GMP/blob/master/mpz/millerrabin.c

VE-5422-2: Generators of $\mathbb{G}_q$ (Resolved)

Summary: Making sure algebraic groups have the expected properties (pg. 18 [7]). For example, drawing a distinction between $\mathbb{G}_q$, which includes 1, and the *generators* of $\mathbb{G}_q$, which excludes 1.

Action Taken: Section 7.3 of the primitives specification (version 1.0.0) now includes explicit text acknowledging this.

VE-5422-3: Increasing the Default Security Level (Unresolved)

Summary: The *default* and *extended* security levels should be increased (pg. 3, 16-17 [7]).

Action Taken: No change.

Remark: The 112-bit "default" security level remains out of line with current NIST and ECRYPT [15,3,6] recommendations for applications with a security lifetime past 2030. Since the use-case is elections, and since 2030 is only two 4-year election cycles away, the security lifetime (esp. ballot secrecy) of election data generated henceforth likely extends past the 2030 horizon.

At a minimum, the Chancellery should be consulted with regard to what this security lifetime should be (in years).

VE-5422-4: Implications of Violated Trust Assumptions (Resolved)

Summary: Additional information should be provided about the implications of broken trust assumptions (pg. 3 [7]). For example, the system specification (version 0.9.6) and the protocol specification (version 0.9.11) discuss the threat model and trust assumptions but do not explicitly spell out what happens if any single assumption is broken.

Action Taken: No significant change.

Remark: This is still a worthy discussion, but the question may be too open-ended to address within this process.

4.7 Progress on Key Recommendations of 2021 Final Report

This section pertains to progress on the key findings made in my Final Report. See: Swiss Post Re-Examination note, VE-5423, Section 2.7, X04.00, April 2022.

VE-5423-1: Under-specified Voter Authentication (Unresolved)
Summary: The question of voter identity and authentication is not fully specified. Swiss Post explained authentication is ultimately out their control, and different cantons take different approaches to the problem (pg. 3 [7]).
Action Taken: The protocol specification (Sec 11.2, version 1.0.0) makes this role slightly more explicit, and suggests their security model is independent of voter authentication.
Remark: A concrete world-view of what voter authentication should look like is still missing.

VE-5423-2: Labeling Trusted Parameters in Algorithms (Resolved)
Summary: Checking the validity of input parameters (pg. 3 [7]).
Action Taken: Unchanged.
Remark: This issue is subsumed by VE-5422-1

VE-5423-3: Response Follow-through (Resolved)
Summary: Only 22 of 137 recommendations from the preliminary (September 2021) report had been addressed in the final (November 2021) report (pg. 3 [7]).
Action Taken: Various improvements have been made.
Remark: Due to time, I have not been able to assess the status of every issue identified in my November 2021 report in this re-examination. However, I would agree Swiss Post has largely and substantively addressed these issues.

VE-5423-4: Integrity Checks of Inputs to the Verifier (Resolved)
Summary: Similar to 5422-1, all inputs to the verifier must be checked to ensure values have the expected form (pg. 4 [7]).
Action Taken: N/A
Remark: This issue is subsumed by VE-5422-1

4.8 Unimplemented Findings

This section pertains to findings Swiss Post has chosen not to implement for the re-examination. See: Swiss Post Re-Examination note, Section 2.2, V01.00, June 2022.

VE-4947: Kleene Star (Resolved)
Summary: Notional position of Kleene star (pg. 13 [7]).
Action Taken: None.
Remark: Stylistic preference.

VE-5116: Subgroup Structure of q (Resolved)
Summary: Are small subgroups of the exponent group $\mathbb{Z}_q^*$ exploitable (pg. 15-16 [7])?
Action Taken: None: "There is no security risk since we rely on DDH. Essex himself writes that this is just an observation that is not immediately exploitable."
Remark: I acknowledge I do not have a pathway to exploit this observation presently, and consider the matter resolved—for now. With that said, I dispute Swiss Post's claim that DDH is the mechanism preventing this risk. The DDH assumption states, in part, that exponents are chosen uniformly and independently in $\mathbb{Z}_q$. My question was about the implications of a specific counter-case; picking exponents from a *subgroup* of $\mathbb{Z}_q^*$, which clearly falls outside the scope of the DDH assumption. It does not, however, fall outside the scope of this protocol. The uniformity of private exponents (a necessary part of the DDH assumption) is not something the participants are required to prove. So the answer to the question "what happens if someone picks exponents in an algebraically sneaky way" cannot be "our security model assumes everyone picks exponents randomly" unless we are (a) working in the *honest-but-curious* model, or (b) there's some kind of verification check. Neither is the case here.

VE-5425: Clarify Double Trust Role (Resolved)
Summary: Are there trust issues with Swiss Post serving a double role as system provider and mail service? (pg. 8 [7]).
Action Taken: "Requirement as stated in the report from Essex is not relevant anymore in the newest version of the OEV. Swiss Post does not envision itself acting fully as the Print Office. We do not print material and the setup is run by the cantons."

4.9 Out of Scope

This section pertains to items Swiss Post considers out of scope of the re-examination. See: Re-Examination note, Section 3, V01.00, June 2022.

VE-5136: Implications of Quantum Computing (Unresolved)
Summary: Provide a detailed description of which mathematical assumptions are broken by an at-scale quantum computer. Describe which specific harms to an election (integrity, ballot secrecy) are likely if a quantum attack occurred against an active or historic election. (pg. 22 [7]).
Action Taken: No action. "Quantum-resistant protocols are not yet mature enough for productive use." The issue will be "explored" in the future.
Remark: This response conflates two separate questions. I was asking about implications of an at-scale implementation of Shor's algorithm against the *existing* protocol. I am not advocating for a major redesign of the protocol around post-quantum primitives. Nor am I optimistic about the prospects of quantum computing. At the time of writing, a real-world implementation of Shor's algorithm has not exceeded factoring 5-bits ($n = 21$), and no direct progress has been publicly reported in the past 10 years.[8] Nevertheless, quantum computing is a credible enough threat that the users of this system should have at least some sense of what the implications are, both to current elections, and to historic election data and cryptographic artifacts. Since the question pertains to the *current system*, I contend it remains in-scope of the re-examination.

4.10 Length-based Attacks

A recent study by Brunet, Pananos, and Essex [5] found that a network-based observer could infer voting intent in some e-voting implementations by examining the length of (encrypted) TLS records exchanged between voter and server.

Although many implementations used fixed-length codes to convey voting intent to the server, length variability in the response was observed in some implementations generating ballot confirmation pages on the sever-side. Swiss Post kindly provided me access to a demo of the voting interface, which allowed me to confirm their implementation is *not* vulnerable to this ballot confirmation page attack owing to page-generation performed on the client-side.

I did not, however, observe any explicit analysis in the protocol specification demonstrating that the protocol is free from (exploitable) length-based variability, so I am flagging it as a topic of future consideration.

[8] https://en.wikipedia.org/wiki/Integer_factorization_records#Records_for_efforts_by_quantum_computers

5 Addendum: Remaining VEs Re-examined

This addendum captures the remaining 30 issues not explicitly responded to by Swiss Post in the re-evaluation documentation.

VE-4928: Correctness IDs (Resolved)
Summary: It was not clear in the system specification how CorrectnessID's prevent an invalid plaintext from being encrypted (pg. 11 [7]).
Action Taken: Additional clarification and examples were provided in Section 3.4.3 in version 1.0.0 of the systems specification.

VE-4963: Length-extension attack resistance of `RecursiveHash` (Resolved)
Summary: How does `RecursiveHash` prevent length-extension attacks (pg. 14 [7])?
Action Taken: System specification (version 1.0.0) explicitly claims "length extension attacks are prevented" by the `RecursiveHash` construction and cites CHVote's `RecHash` function [9], and the primitives specification specifies SHA-3 as the underlying hash function.
Remark: A length-extension attack on a cryptographic hash function $h : \{0,1\}^* \to \{0,1\}^\ell$ is one in which $h(x_1 \| x_2)$ could be efficiently computed given a hash image $h(x_1) \in \{0,1\}^\ell$ and a pre-image $x_2 \in \{0,1\}^*$. According to its designers, SHA-3 is not susceptible to length-extension attacks.[9]

VE-4971: Smallest generator (Unresolved)
Summary: What is the justification for picking the smallest integer $x > 1$ that is a generator of $\mathbb{G}_q$ in `GetEncryptionParameters` in the primitives specification (pg. 15 [7])?
Action Taken: No change.
Remark: As previously pointed out in the Final Report, this seems to offer no security relative to e.g., always picking 4. There are two possibilities: Assume picking the smallest generator makes no difference to security. Then the approach can be simplified to always pick 4 and the "smallest generator" algorithm can be safely eliminated. Conversely, assume picking the smallest generator *does* is important to the underlying computational hardness assumption. However, because the `GetEncryptionParameters` algorithm in the primitives specification only leads to *two* possible outcomes (2 or 4), and because the chosen generator is a function of the input `seed` value, the election agency can trivially induce a generator of their choosing by tweaking the `seed` and re-running the algorithm until it returns the chosen value, circumventing the assumed contribution to the hardness assumption.

[9] https://keccak.team/keccak_strengths.html

VE-5117: Testing parameters (Unresolved)

Summary: The testing-only security level is trivially weak and would violate several important security assumptions in the event of accidental deployment (pg. 16 [7]).

Action Taken: Test parameters were changed to $|p| = 8n$ for $n \geq 1$. The minimal valid test parameters remain trivially weak, increased from $p = 5, q = 2$ to $p = 167, q = 83$, representing a group size of $|q| = 7$, offering 4-bits of security.
In the *Response to Examination Report by Aleksander Essex – Scope 1 Cryptographic Protocol*, Swiss Post states:

> *These trivial parameters are highly useful in validating work during development and testing. We keep this testing-only level, because we have other safeguards in place, notably the independent auditors verify the configuration phase. They ensure the cryptographic parameters are safe and have been verifiably chosen.*

Remark: What safeguards? The safeness of prime p and the verifiability of the other parameters are necessary but not sufficient. $p = 167$ is a valid possible output of `GetEncryptionParameters` given the minimally allowable $\lambda = |p| = 8$, yet is still trivially weak. Nothing in the verifier specification (incl. in `VerifyEncryptionParameters()`) appears to direct the verifier to check the bit-lengths of the parameters provided in `encryptionParametersPayload.json` to ensure they conform the *standard* or *extended* security levels. Nothing evidently appears to programmatically preclude accidental deployment of weak parameters.

VE-5118, 5119: Elgamal edge cases (Resolved)

Summary: The (possibly problematic) triviality of exponentiation of the form $1^x \bmod p$, in which 1 is *not* a generator of $\mathbb{G}_q$ (pg. 18 [7]).

Action Taken: Swiss Post added discussion of Section 7.3 Key Pair Generation (primitives specification version 1.0.0) specifically addressing this issue.

VE-5123: Clarify design choice of Bayer-Groth mixnet (Unresolved)

Summary: Why was the Bayer-Groth verifiable mixnet architecture chosen (pg. 19 [7])?

Action Taken: No change.

Remark: Clarify if there are any particular issues that bind Swiss Post to this architecture, such as patent encumbrance. How does the speed and algorithmic complexity compare to, say, Verificatum?[10] Were any alternative architectures considered?

[10] https://www.verificatum.org

VE-5124: Unsupported claim (Unresolved)
Summary: Swiss Post claims verifiable mixnets "underpin most modern e-voting schemes" (pg. 19 [7]). A systematic literature review should be cited to support this claim.
Action Taken: No change.

VE-5126: Implied modular reduction in subscript (Unresolved)
Summary: Algorithm 5.4 (`GenPermutation`) specifies an offset value for a subscript $\pi_{i+offset}$. The implied modular reduction by the permutation of size N should be stated ,e.g., in the summary text directly above. (pg. 19 [7]).
Action Taken: No change.

VE-5129: Unclear vector notation (Resolved)
Summary: Unclear vector notation in Algorithm 6.8 Line 7 in the primitives specification, in version 0.9.5[11] (pg. 21 [7]).
Action Taken: Clarified in Algorithm 9.11 in version 1.0.0.

VE-5131–5135: Ensuring inputs have expected cryptographic form (Resolved)
Summary: Explicit cryptographic tests for algebraic group membership (pgs. 21–22 [7]).
Action Taken: Subsumed by VE-5422-1.

VE-5424: Integrity verification in the verifier specification (Resolved)
Summary: Ensuring data elements "correspond to the specification" and are "all within the specified ranges" (pg. 23 [7]).
Action Taken: Subsumed by VE-5422-1.

VE-5428: Improper quotation marks (Unresolved)
Summary: Instances of quotation of the form "quote" should be corrected to "quote" (pg. 23 [7]). See e.g. "`GenEncLongCodeShares`" in `VerifyEncryptedPCCExponentiationProofsVerificationCardSet()` in the verifier specification.
Action Taken: No change.

[11] My original report incorrectly referenced this as Algorithm 6.7.

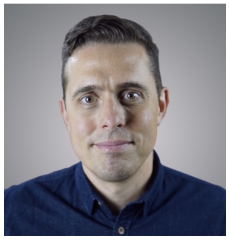
VE-5429–5440: Various issues with secure logs in verifier specification (Resolved)
Summary: Various notation and algorithm focused remarks relating to the specification of secure logs in Section 4.3 of the verifier specification (version 0.9.1) (pgs. 23–24 [7]).
Action Taken: Mention of secure logs no longer appears in version 1.01.
Remark: I was not able to determine why this section was removed.

VE-5421: Additional clarity needed on voter authentication (Unresolved)
Summary: Would attacking the mail system and stealing the voter code sheet package provide sufficient information to cast a vote on a voter's behalf (pgs. 8–9 [7])?
Action Taken: Swiss Post responded in the JIRA Ticket Inventory List by pointing out that "the channel from the Print Office to the Voter (through Swiss Post) is trustworthy in the trust assumptions of the OEV (2.10.2)" and that "the date of birth was a requirement set by the FCh."
Remark: I have consulted the OEV and can seem to find no direct reference to a date of birth requirement. However, regardless of what the OEV requires, I hope we can at least agree that an envelope delivered to a voter by postal mail should probably not, on its own, be sufficient cast a ballot—*even if* you assumed the communication channel was secure. For example, there was a recent case in Canada of some municipal voters throwing their e-voting code sheet envelopes into a garbage bin at their local post office. Errors in the voter registry had led to some voters being disenfranchised in the previous election cycle, which led this municipality to eliminate date of birth in the voter authentication process. Here simply *seeing* the contents of the voter code sheet would be sufficient to maliciously cast a ballot on the voter's behalf. Notably, even with the OEV's trust assumption about the transmission channel being perfectly secure, in this scenario the voter's ballot is *still* susceptible to fraud in a realistic setting. It remains unclear to me what the precise voter authentication requirements look like. What exactly is necessary (if anything) beyond the printed code sheet to cast a ballot on another voter's behalf? Swiss Post provided me access to an online demo (at my request) in February 2022, allowing me to cast a ballot as a test voter. It required a 20-character initialization code and a 4-digit birth year (i.e., this was the information sufficient to cast a ballot). So maybe as a concrete question: Is this representative of the envisioned real-world deployment?

References

1. M. Albrecht, J. Massimo, K. Paterson, and J. Somorovsky. Prime and Prejudice: Primality Testing Under Adversarial Conditions. In *Cryptology ePrint Archive, Report 2018/749*, 2018.
2. R. Baillie and S. S. Wagstaff. Lucas pseudoprimes. *Mathematics of Computation*, 35:1391–1417, 1980.
3. E. Barker and A. Roginsky. Transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication 800-131A Rev. 2*, 2019.
4. D. Basin. Review of Symbolic Proofs for Swiss Post's Voting System. In *Report Submitted to the Swiss Federal Chancellery*, 2021.
5. J. Brunet, A. D. Pananos, and A. Essex. Review your choices: When confirmation pages break ballot secrecy. In *International Joint Conference on Electronic Voting*, pages 36–52, 2022.
6. M. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. *NIST Special Publication 800-38-D*, 2007.
7. A. Essex. Analysis of the Swiss Post e-Voting System. Audit Scope 1: Cryptographic Protocol. In *Report Submitted to the Swiss Federal Chancellery*, 2021.
8. B. Ford. Auditing the Swiss Post E-voting System: An Architectural Perspective. In *Report Submitted to the Swiss Federal Chancellery*, 2021.
9. R. Haenni, R. Koenig, P. Locher, and E. Dubuis. Chvote protocol specification, version 3.2. In *IACR e-print archive*, 2020.
10. R. Haenni, R. Koenig, P. Locher, and E. Dubuis. Examination of the Swiss Post Internet Voting System. Scope 1: Cryptographic Protocol. In *Report Submitted to the Swiss Federal Chancellery*, 2021.
11. T. Haines, O. Pereira, and V. Teague. Report on Swiss Post e-Voting System. In *Report Submitted to the Swiss Federal Chancellery*, 2021.
12. G. L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
13. C. Pomerance, J. L. Selfridge, and S. S. Wagstaff. The pseudoprimes to $25 \cdot 10^9$. *Mathematics of Computation*, 35:1003–1026, 1980.
14. M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.
15. N. Smart et al. Algorithms, key size and protocols report (d5.4). *ECRYPT-CSA H2020-ICT-2014 – Project 645421*, 2018.

A Author Bio



Aleksander Essex is an associate professor of software engineering at Western University in Canada in the Department of Electrical and Computer Engineering. His research specializes in cybersecurity and applied cryptography, with a focus on voting technology and evidence-based elections. He has numerous publications on e-voting security and cryptographic end-to-end election verification (E2E-V). He sits on several governmental advisory and standards development committees on election technology and has done over one hundred press interviews in Canada on the topic.