

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

Examination of the Swiss Post Internet Voting System

Releases 1.5.0 (June 2025), 1.5.1 (September 2025), 1.5.2 (October 2025),
1.5.2.1 (November 2025) and 1.5.2.2 (December 2025)

Rolf Haenni, Reto E. Koenig, Philipp Locher

Final Version

January 9, 2026

On behalf of the Federal Chancellery

Contents

1. Introduction	3
1.1. Purpose and Goals of Mission	4
1.2. Topics of Examination	5
1.3. Overview of Releases 1.5.0, 1.5.1, 1.5.2, 1.5.2.1, 1.5.2.2	6
1.3.1. Specification Documents	6
1.3.2. Source Code	7
1.3.3. Overview of Changes	7
1.3.4. Status Overview on Jira	8
2. Summary of Findings	10
2.1. Crypto-Primitives	10
2.1.1. Discussion of Measure A.15	11
2.1.2. Discussion of Measure A.16	13
2.2. E-Voting System	13
2.3. Voting-Card-Print-Service and PDF-Verification-Service	27
A. Changelogs	30
A.1. Crypto-Primitives	30
A.2. Data Integration System	33
A.3. E-Voting	34
A.4. E-Voting Libraries	37
A.5. Voting-Card-Print-Service and PDF-Verification-Service	39
A.6. Verifier	41

1. Introduction

This examination report discusses the findings of our analysis of the Swiss Post e-voting system, which we conducted between June and December 2025. It is a continuation of previous reports that we submitted to the Federal Chancellery (FCh) between March 2022 and July 2024. All existing reports are publicly available on the FCh’s web site, together with reports from other experts.¹ While this document is largely self-contained, it may still be necessary to look up earlier discussions on certain subject matters for achieving a better understanding of their exact context. Whenever useful or necessary, we provide pointers to the reports from the following list of references:

[BFH22a] Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. *Examination of the Swiss Post Internet Voting System – Scope 1 (Cryptographic Protocol)*, March 28, 2022.

[BFH22b] Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. *Examination of the Swiss Post Internet Voting System – Scope 2 (Software)*, March 28, 2022.

[BFH23a] Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. *Re-Examination of the Swiss Post Internet Voting System – Scope 1 (Cryptographic Protocol) and Scope 2 (Software)*, Version 1.0.2, February 23, 2023

[BFH23b] Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. *Re-Examination of the Swiss Post Internet Voting System — Releases 1.2.3 (February 2023) and 1.3 (April 2023), with Addendum on Version 1.3.1*, June 30, 2023.

[BFH23c] Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. *Examination of the Swiss Post Internet Voting System — Releases 1.3.2 (July 2023) and 1.3.3 (October 2023), with Addendum on Sub-Release 1.3.3.2*, December 22, 2023.

[BFH24] Rolf Haenni, Reto E. Koenig, Philipp Locher. *Examination of the Swiss Post Internet Voting System — Releases 1.4.0 (February 2024) and 1.4.1 (March 2024), (with Addendum on Releases 1.4.2, 1.4.3, 1.4.3.1)*, July 25, 2024.

Many of the findings and issues discussed in these reports have been addressed in corresponding updates released during the year of 2024 or earlier. For keeping track of the findings and their updates, the issue tracking platform Jira has been set up by the Federal Chancellery in 2024. Most issues that have been resolved to the satisfaction of all participants have been closed (see Table 2 and [BFH24, Annex C]).

One of the goals of this document is to recapitulate the new issues reported in the 2025 examination period for the four releases obtained between June and December 2025. While doing the examination, the reporting of the findings on Jira has been a continuous and transparent process.

A first and a second draft of this document has been sent to the Federal Chancellery on November 9 and December 12, 2025, respectively, and the pre-final and final versions were submitted on December 17, 2025, and on January 9, 2026, respectively. The examination and the writing of this report have been conducted jointly by the listed authors from the Bern University of Applied Sciences and independently of any other group of people.

¹See https://www.bk.admin.ch/bk/de/home/politische-rechte/e-voting/ueberpruefung_systeme.html

1.1. Purpose and Goals of Mission

We have been assigned by the Federal Chancellery with this supplementary examination task on December 5, 2024. We received a detailed release and examination schedule for the whole year of 2025 (see Figure 1), which included various deadlines for submitting several drafts and the final version of this document. The publication of all examination reports is scheduled for January 2026.

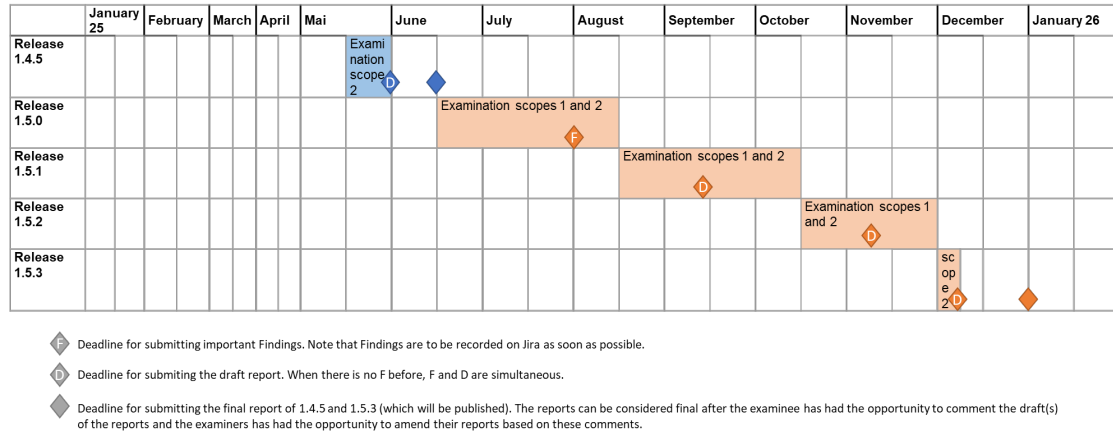


Figure 1: Release and examination schedule 2025

In an online kick-off meeting held on July 17, 2025, with all participating experts and representatives from both the Federal Chancellery and Swiss Post, we obtained more comprehensive information regarding the upcoming releases and our task to review them. The updates of the Swiss Post Voting System were released following the schedule shown in Figure 1. Shortly after each release date, email announcements were sent to all involved experts containing additional information regarding the implemented changes, links to corresponding changelog documents on gitlab.com, and filters for obtaining condensed lists of affected Jira issues. The release dates and the dates of the email announcements are shown in Table 1.

Release date	Announcement	Description
20.06.2025	24.06.2025	Publication of Release 1.5.0 (Major release)
11.07.2025	11.07.2025	Publication of Release 1.5.0.1 (Minor technical patch)
15.08.2025	18.08.2025	Publication of Release 1.5.1 (First patch, bug fixes from the first round)
10.10.2025	13.10.2025	Publication of Release 1.5.2 (Second patch, bug fixes from the first and second rounds)
28.11.2025	–	Publication of Release 1.5.3 (Production version, maintenance patch of release 1.5.2)

Table 1: Release schedule 2025

The purpose and goals of our examination are defined in a document from the the Federal Chancellery called “*Audit Concept*”. This document is based on the legal requirements as defined in the current “*Federal Chancellery Ordinance on Electronic Voting*” (OEV)

and its annex from May 2022 [BK22]. Earlier in 2025, on February 17, we received an updated Version 1.6 of the Audit Concept document [BK25].

As in earlier missions, our current assessment is limited to Scope 1 (Analysis of Cryptographic Protocol) and Scope 2 (Software). As in the precedent document [BFH24], we report on both scopes simultaneously in a single document. This allows us to better exploit the strong interconnection between Scope 1 and Scope 2 in many topic areas that need to be examined. For the sake of completeness, we quickly recite some statements from the Audit Concept that are most relevant for defining the purpose and goals of our assessment:

In the context of the assessment of the Swiss Post system, the experts shall answer the following questions:

- Are the system, its development and operation compliant with the legal requirements?
- Are the measures taken to mitigate risks effective?
- Which improvements could be made for the sake of security, trust and acceptance?

The following chapters describe the different scopes [...]. Together, they cover the entire system, its development and operation. [...]

- Scope 1 (Cryptographic Protocol): The protocol must fulfil the requirements listed in number 2 of the annex of the OEV.
- Scope 2 (Software): The software of the system including the auditor’s technical aid must fulfil the requirements listed in numbers 2 to 25 of the annex of the OEV and adequately support the protocol (number 2 of the annex of the OEV). The mapping between a requirement in those paragraphs and the place (functionality in the system, organisational procedure, element of infrastructure, etc.) where it is fulfilled shall be provided by the examinees before the examination. If the mapping table indicates that a requirement is covered in another scope than the one specified here, the examiner shall verify this claim. Functions whose trustworthiness is decisive for the effectiveness of verifiability as per OEV, must be examined in detail on the basis of the source code and the cryptographic protocol.

1.2. Topics of Examination

Another important document from the Federal Chancellery is the “*Catalogue of Measures*” from August 2023 [BK23]. This document recapitulates areas requiring action in relation to certain non-conformities, as well as areas where further improvements are necessary to guarantee better compliance with the criteria as defined in the OEV. It includes a list of measures for the following four categories of measures:

- A. Further development of the systems (Measures A.1 to A.26)
- B. Effective control and oversight (Measures B.1 to B.14)
- C. Increasing transparency and trust (Measures C.1 to C.8)
- D. Closer cooperation with the academic community (Measures D.1 to D.3)

For each of the pending measures, the implementation status is indicated as *planned*, *in progress*, or *ongoing*. The category of measures that is most relevant for this assessment is the first one on the further development of the system (A.1 to A.26). In the annex of [BK23], some additional information is provided for the pending measures A.9, A.11, A.13, and A.22. The whole catalogue defines the general guidelines for the current and future progression of the project over the next few years.

In the instructions given for the current examination, experts were asked to enter all new issues directly into the Jira platform. In their announcement of Release 1.4.1 in March 2024, Swiss Post started to include pointers to these issues in their list of changes. They continued to do so in the announcements of the Releases 1.5.0 to 1.5.2. Within the Federal Chancellery’s effort to direct the future system development in a more systematic manner, obtaining these pointers is very beneficial for examiners to efficiently assign the announced changes to the affected Jira issues.

During the course of our assessment, a total of 27 new issues were entered on the Jira platform by the authors of this report. We agreed with the Federal Chancellery to include these issues in this report without giving further comments or explanations (see Section 2). The nature of this document is therefore somewhat distinct in comparison to previous examination reports. At the time of writing this report, 3 of the reported issues are labeled as *delivered*, 13 as *clarified*, and 11 as *planned* (see Tables 2 and 3). Note that another 62 issues from the 2024 examination period remain labeled as *planned* and 13 as *clarified*. For further details on these unsolved issues, we refer to the summary given in [BFH24, Annex C].

1.3. Overview of Releases 1.5.0, 1.5.1, 1.5.2, 1.5.2.1, 1.5.2.2

1.3.1. Specification Documents

To conduct our examination of the versions released in 2025, we downloaded the most recent specification documents from Swiss Post’s public repositories on gitlab.com.² Some documents remained unchanged since Release 1.5.0 (published in June 2025), except for [SysSpec] and [CryptPrim], which have been slightly modified in Release 1.5.1 (published in August 2025) and in Release 1.5.2 (published in October 2025).

- [SysSpec] *Swiss Post Voting System – System Specification*, Version 1.5.2, Swiss Post Ltd., October 10, 2025
- [CryptPrim] *Cryptographic Primitives of the Swiss Post Voting System – Pseudo-code Specification*, Version 1.5.1, Swiss Post Ltd., August 15, 2025
- [VerSpec] *Swiss Post Voting System – Verifier Specification*, Version 1.6.0, Swiss Post Ltd., June 20, 2025
- [ProtProofs] *Protocol of the Swiss Post Voting System – Computational Proof of Complete Verifiability and Privacy*, Version 1.4.0, Swiss Post Ltd., June 20, 2025
- [ArchDoc] *E-Voting Architecture Document*, Version 1.5.0, Swiss Post Ltd., June 20, 2025

As in earlier releases, special versions of these documents were given to the examination experts with all the changes highlighted directly in the documents (except for [ArchDoc]), and summaries of the changes are available in the documents’ revision charts and the CHANGELOG.md files of corresponding repositories on gitlab.com.

²See <https://gitlab.com/swisspost-evoting>

1.3.2. Source Code

The announced software releases were published according to the timeline from Table 1. Specifically, Release 1.5.0 was published on June 16, Release 1.5.1 on August 15, and Release 1.5.2 on October 10, 2025. On November 20 and December 8, two *maintenance patches* (Releases 1.5.2.1 and 1.5.2.2) were released instead of the announced *production version* (Release 1.5.3).

As in previous releases, the version numbering of the system components are not fully consistent with the top-level release numbering: the latest versions of the **verifier** component have been released as Versions 1.6.0, 1.6.1, 1.6.2, and 1.6.2.1, and the latest version of the **data-integration-service** component as Versions 2.9.0, 2.9.1, 2.9.2, 2.9.2.1, and 2.9.2.2 respectively. Similar to previous reports, we generally refer to the latest software versions as the *June 2025 release*, *August 2025 release*, *October 2025 release*, *November 2025 release*, and *December 2025 release*.

1.3.3. Overview of Changes

In the above-mentioned e-mail announcement on June 24 for the June 2025 release, the modifications made to both the specification documents and the code were not explicitly described. However, by incrementing the versioning to the next minor number, i.e., from 1.4.3 to 1.5, it is evident that a significant number of modifications have been implemented throughout the entire system.

From the point of view of the system's general architecture, the most visible change is the inclusion of two additional components called **voting-card-print-service** (VCPS) and **PDF Verification Service** (PVS). The former is a standalone application that generates voting cards from e-voting system input files using canton-specific layouts and configurations. It allows customization of layouts and text for each voting event to meet corresponding requirements. The application produces digitally signed and symmetrically encrypted PDF voting cards for printing and mailing to voters. The encryption password is sent by the canton to the print office via an out-of-band channel, and the print office uses the PDF Verification Service to decrypt and verify the files before printing. VCPS 3.0 from December 2024 has been examined by other external experts. Their report is publicly available [OH25]. Corresponding improvements have already been implemented in subsequent Versions 3.1, 3.2, and 3.3. An update to version VCPS 3.3.1 has been released along with the June 2025 release of the system (together with PVS 1.5), and the latest version VCPS 3.3.2.1 has been released along with the November 2025 release. PVS has been further updated to 1.5.1, 1.5.2, and 1.5.2.1, respectively, in the August, October, and November 2025 releases.

For the modifications made to the core components of the system such as **crypto-primitives**, **crypto-primitives**, **e-voting**, **e-voting-libraries**, or **verifier**, we refer to the **CHANGELOG.md** files in corresponding repositories on **gitlab.com**, where relevant changes have been reported accurately and in a systematic manner (see Subsections A.1 to A.6). In the majority of cases, the modifications consist of modest optimizations, small enhancements, or minor bug fixes. Using the pointers given in corresponding statements, locating the relevant code lines is relatively straightforward. In comparison with the major release from June 2025, the August, October, November, and December 2025 releases contain

much smaller amounts of modifications (the first two were announced as *first* and *second patch*, respectively, see Table 1).

1.3.4. Status Overview on Jira

In their email announcements, Swiss Post also supplied direct pointers to the affected issues on the Jira platform. Additionally, they modified the status of the issues that corresponded to those affected issues to either *clarified*, *planned*, or *delivered*. Note that in this context, *clarified* merely indicates that Swiss Post has responded to the issue, not that the issue itself has been fully resolved, or that the reporters are satisfied by the clarification. In Table 2, we give an overview of all 106 issues reported by the authors of this document, which at the time of writing this report can be considered as *resolved* (*abandoned*, *delivered*, *closed*). Most of them have been reported in the 2024 report [BFH24] and resolved in versions earlier than the ones currently under examination.

Reported	Issues	Status	Release
[BFH24]	EX-64, EX-67, EX-69, EX-74, EX-76, EX-106, EX-107, EX-134, EX-227	abandoned	
	EX-63, EX-65, EX-66, EX-88, EX-93, EX-98, EX-105, EX-124, EX-127, EX-128, EX-129, EX-141, EX-142, EX-145, EX-146, EX-148, EX-149, EX-151, EX-153, EX-154, EX-164, EX-175, EX-182, EX-195, EX-197, EX-198, EX-209, EX-218, EX-219, EX-220, EX-235, EX-243, EX-244, EX-246, EX-247, EX-249, EX-250, EX-251	closed	
	EX-8, EX-9, EX-10, EX-11, EX-13, EX-61, EX-84, EX-113	delivered	≤1.4.5
	EX-62, EX-73, EX-75, EX-77, EX-82, EX-83, EX-92, EX-94, EX-99, EX-104, EX-108, EX-109, EX-110, EX-112, EX-115, EX-118, EX-125, EX-126, EX-131, EX-133, EX-137, EX-138, EX-139, EX-140, EX-143, EX-144, EX-147, EX-150, EX-155, EX-157, EX-158, EX-159, EX-168, EX-170, EX-172, EX-178, EX-179, EX-180, EX-183, EX-184, EX-186, EX-187, EX-188, EX-200, EX-201, EX-202, EX-204, EX-206, EX-207, EX-212, EX-221, EX-222, EX-223, EX-241, EX-260, EX-263	delivered	1.5.0
[this report]	EX-332, EX-333	delivered	1.5.0
	EX-398	delivered	1.5.2

Table 2: Issues currently considered as resolved on the Jira platform.

In Table 3, we give a similar overview of the issues that are currently considered as *unresolved* (*clarified*, *planned*, *suspended*). Along with the 22 unresolved issues from this

examination period (11 clarified, 11 planned), there is still a large number of unresolved issues from the 2024 examination period (13 clarified, 62 planned, 2 suspended). Most of them are marked as [to be] “*fixed in Release 2.0*”. These are references to the next major release announced for 2026 or 2027 implementing a fully re-designed cryptographic protocol that looks very different in comparison with the current one (an updated white paper describing this protocol has been given to us on December 1, 2025). To the best of our understanding, the decision for re-designing the underlying cryptographic protocol results from fundamental difficulties of implementing some of the items listed in the Federal Chancellery’s catalogue of measures [BK23] for the given protocol. Some of these measures are closely related to issues listed as unresolved in Table 3.

Reported	Issues	Status
[BFH24]	EX-12, EX-89, EX-100, EX-132, EX-135, EX-136, EX-161, EX-173, EX-174, EX-190, EX-191, EX-192, EX-203	clarified
	EX-68, EX-70, EX-71, EX-72, EX-78, EX-79, EX-80, EX-81, EX-90, EX-91, EX-95, EX-96, EX-97, EX-101, EX-102, EX-103, EX-114, EX-116, EX-117, EX-156, EX-160, EX-162, EX-163, EX-165, EX-166, EX-167, EX-169, EX-171, EX-176, EX-177, EX-181, EX-185, EX-189, EX-193, EX-194, EX-196, EX-199, EX-205, EX-208, EX-210, EX-211, EX-213, EX-214, EX-215, EX-216, EX-217, EX-224, EX-225, EX-228, EX-229, EX-230, EX-231, EX-232, EX-233, EX-234, EX-236, EX-237, EX-238, EX-239, EX-240, EX-242, EX-245	planned
	EX-7, EX-152	suspended
[this report]	EX-385, EX-390, EX-392, EX-393, EX-394, EX-397, EX-399, EX-401, EX-402, EX-410, EX-411, EX-414, EX-415	clarified
	EX-383, EX-384, EX-386, EX-387, EX-388, EX-389, EX-391, EX-395, EX-396, EX-400, EX-403	planned

Table 3: Issues currently considered as unresolved on the Jira platform.

2. Summary of Findings

As the reporting of findings has entirely shifted from listing them in the examination reports to entering them on the Jira issue tracking platform, we agreed with the Federal Chancellery to no longer include separate discussions of our findings in this document. Because Jira is transparent to all stakeholders from Swiss Post and the Federal Chancellery, discussions with the expert are directly conducted on the platform. For some of the issues that we entered during the 2025 examination period, we have already received feedback and clarifications from Swiss Post. These discussions are an ongoing process and are expected to continue in the aftermath of writing this report. The report itself can therefore be seen as a snapshot of the progress accomplished thus far based on the reports from the all involved experts. What is important is the fact, that this process is now carried out in a systematic manner using a tool that is available to everyone and as such prevents that something gets forgotten.

2.1. Crypto-Primitives

Considering the Jira issues in Table 2 with status updated to *delivered* in Release 1.5.0, quite some changes have been implemented to the **crypto-primitives** component (all issues listed between EX-133 and EX-241). Many of the changes are responses to the following general complaints from the 2024 examination report [BFH24, Sections 2.1.1–2.1.4]:

- Principle of Generality,
- Formal Notation,
- Algorithmic Simplicity,
- Software Design.

Looking at the implemented changes in both the specification document and the component’s source code (see corresponding changelogs in Subsection A.1), we acknowledge that great efforts have been spent by Swiss Post to address our findings in a systematic way. Moreover, we observed that minor problems encountered in Version 1.5.0 of the specification document [CryptPrim] have been corrected in Version 1.5.1.

As part of our current examination of the Releases 1.5.0 to 1.5.2.2, we have inspected the changes made to the **crypto-primitives** component. Besides the issues that are still unresolved (see Table 3 and the discussion in Subsections 2.1.1 and 2.1.2), we encountered only three minor problems and entered them as issues EX-332, EX-414, and EX-415 on the Jira platform. While EX-332 has already been fixed in Version 1.5.0, we anticipate that EX-414 and EX-415 will be fixed in an upcoming release because addressing them is quite simple.

EX-332: Alg. 3.10 IntegerToFixedLengthByteArray

Description

In der Spezifikation sehen wir eine Vereinfachung in Zeile 5, indem der erste Loop (Zeilen 2–4) weggelassen und stattdessen im zweiten Loop (Zeile 5) die Bedingung von “**for** $i \in [0, m)$ **do**...” auf “**for** $i \in [0, n)$ **do**...” erweitert wird. Dies erlaubt dann auch das

Weglassen der Zeile 1. Des Weiteren kann nun Algorithmus 3.9 vereinfacht werden, indem dieser neu Algorithmus 3.10 mit dem zweiten Parameter $n \leftarrow \text{ByteLength}(x)$ aufruft.

Related Issues

EX-23 (in progress), EX-334 (closed)

Created

02/06/2025

Status

Delivered (fixed in Release 1.5.0)

EX-414: Crypto-Primitives Alg. 5.3 GenRandomString

Description

The size k of the alphabet \mathbb{A} must be strictly greater than 0, because `GenRandomInteger` requires an input integer $m > 0$. The constraint is missing in both the specification and the implementation, for example in the class `Alphabet`.

Recommendation

Add the constraint $k > 0$ to the specification and wherever needed in the code.

Created

09/11/2025

Status

Clarified

EX-415: Crypto-Primitives Algorithms 8.5, 8.9, and 8.10

Description

Algorithm 8.9 `GetMessage` and 8.10 `GetPartialDecryption` have been generalized to support the edge case $\ell = 0$, but Algorithm 8.5 `GetCiphertext` explicitly excludes this edge case (same problem in implementation).

Recommendation

Make the three algorithms consistent with respect to the edge case $\ell = 0$.

Created

09/11/2025

Status

Clarified

2.1.1. Discussion of Measure A.15

The Swiss Post system applies design principles intended to improve maintainability and reduce errors, i.e., its cryptographic primitives are implemented with object-oriented

methods. We concluded in an earlier report that the system could benefit from a more consistent and rigorous use of object-oriented design principles, particularly in clearer naming, cleaner class hierarchies, and more appropriate higher-level abstractions [BFH23a, Section 2.3.1, Pages 44–47]. Measure A.15 from [BK23, Page 7] particularly calls for the design principles of object-oriented programming to be better applied to the protocol’s underlying mathematics, for example to algebraic groups, tuples, vectors, matrices, and sets.

While it is true that certain details have been improved as a response to Measure A.15, we think that the potential for further improvements is still very high in areas such as the ones mentioned above. One particular area of problems was related to the class hierarchy shown in [BFH23a, Page 44]. In the current version of the December 2025 release, this hierarchy looks slightly different (for example `PrimeGqElement` extending `GqElement` and `GqElement` extending `GroupElement`, instead of extending `MultiplicativeGroupElement`), but most problems from the earlier class hierarchy still exist. One such problem is the implementation of the interface

- `GroupVectorElement` (“*Elements of a GroupVector or GroupMatrix*”)

by three classes

- `GroupElement` (“*Representation of a mathematical group element*”),
- `GroupVector` (“*Represents a vector of non-null {@link GroupElement}s ...*”),
- `GroupMatrix` (“*Represents a matrix of {@link GroupVectorElement} elements ...*”),

which we consider poor design for the same reasons as in the version examined in 2023. For example, if a `GroupVector` is a vector of `GroupElement`’s as stated in the class’s JavaDoc, it should not contain a list of `GroupVectorElement`’s (*has*-relationship) and simultaneously inherit from `GroupVectorElement` (*is*-relationship). Almost all of our remarks from the section on “*Implementation of Mathematical Groups*” [BFH23a, Page 44] are still applicable to the version available today.

The same holds for our remarks in the section on “*Implementation of Tuples, Vectors, and Matrices*” [BFH23a, Page 45]. The aforementioned classes `GroupVector` and `GroupMatrix` provide basic implementations of vectors and matrices in a mathematical sense; nevertheless, their limitation to elements of type `GroupVectorElement` greatly diminishes their generality. Furthermore, generic implementations of tuples (pairs, triples, quadruples, etc.) are still entirely missing (see [ArchDoc, Section 9.3.8]³), despite their huge potential of simplifying the code related to the cryptographic protocol at numerous places. The same holds for corresponding sets and Cartesian products, which are very useful for writing lightweight and efficient set membership tests in the implementation of the protocol’s algorithms.

³Swiss Post justifies their decision to refrain from implementing generic tuple classes by giving an example of a model class `ProductStatement`, which in mathematical terms is a quadruple (four elements of different sets). By extending a generic quadruple class, for example by `ProductStatement extends Quadruple<GroupVector<GqElement, GqGroup>, ZqElement, Integer, GqGroup>`, the resulting model class would inherit all pre-defined features from its super class for free, for example immutability, type-safety, non-null checks, consistent implementations of `equals`, `hashCode`, and `toString`, hashability, serializability, streamability, etc. Since there are many model classes similar to `ProductStatement`, the benefit from delegating these features to common super-classes is huge, but Swiss Post appears to be oblivious to the unexploited possibilities.

As a guideline for implementing Measure A.15 more appropriately, we refer to the OpenCHVote module `ch.openchvote.base.utilities`⁴ and its application to the protocol’s algorithms in the module `ch.openchvote.core.algorithms`.⁵

2.1.2. Discussion of Measure A.16

According to Measure A.16 [BK23, Page 7], the system’s protocol and underlying algorithms should be kept as simple as possible so that errors can be detected and corrected early, even though secure e-voting systems inevitably require a certain level of complexity. Swiss Post’s system is not unnecessarily complicated in every respect, but there are still opportunities to simplify it further without compromising security or undermining reasonable design constraints, as also noted in earlier reports [BFH23a, BFH24]. Especially with respect to the examples listed under “*Algorithmic Simplicity*” [BFH24, Section 2.1.3] and “*Potential for Simplifications*” [BFH23a, Section 1.4], respectively, there is quite some room for further improvements. A prominent example is our recommendation of simplifying the Bayer-Groth shuffle proof parameters to $n = N$ and $m = 1$, which offers enormous potential for simplifications in both the specification and the code. However, we are aware that corresponding Jira issues, for example EX–216, are currently marked as [to be] “fixed in Release 2.0”. While we regret Swiss Post’s decision to postpone certain topics that seem important to us, we understand that their priorities might be different from ours.

2.2. E-Voting System

We also reviewed the specification and implementation of the e-voting and e-voting-libraries components based on the changes made for the June, August, and October 2025 releases. The provided PDF files highlighting the modifications to the previous versions and corresponding changelogs facilitated a straightforward review of the specification and the source code (see Subsections A.2 and A.3).

We have documented all our findings in 24 issues on the Jira platform. While we did not identify any significant security problems, we are concerned about the number of inconsistencies, ambiguities, and misalignments added to the new versions. In the light of the large number of unresolved issues from the 2024 examination period (see Table 3), our general reservations from earlier reports are still in place.

EX–333: `LVCKS`

Description

In der System Spezifikation gibt es eine Variable `LVCKS` die als “*Character length of the Base64 encoded verification card keystore*” beschrieben wird. Es findet sich jedoch keine genaue Beschreibung, wie gross dieser Wert ist, oder wie dieser Wert allenfalls hergeleitet werden kann. In der Implementierung wurde dieser Wert als “Magic Number” in der Version 1.4.5 eingeführt: `private static final int keystoreSize = 572`, ebenfalls ohne jegliche weiteren Erläuterungen, wie dieser Wert zustande gekommen ist.

⁴See <https://gitlab.com/openchvote/cryptographic-protocol/-/tree/master/base/utilities>

⁵See <https://gitlab.com/openchvote/cryptographic-protocol/-/tree/master/core/algorithms>

Related Issues

EX-23 (in progress)

Created

02/06/2025

Status

Delivered (fixed in Release 1.5.0)

EX-383: Domain Actual Voting Options**Description**

In the System Specification 1.5.1, Section 3.5.2, the domain of the actual voting option is now correctly specified. However, in the remainder of the specification—particularly within the algorithms—the actual voting option is still defined as an element of the domain \mathcal{T}_1^{50} , which is clearly incorrect. Swiss Post justifies this choice as being “for simplicity”. However, we believe it does not simplify the specification; rather, it introduces confusion and inconsistency. Whether being an implementer or an examiner, one must constantly keep in mind that when a value is specified to be from the domain \mathcal{T}_1^{50} in an algorithm, it might actually not be.

Recommendation

Introduce a new symbol for the domain of the actual voting options, for example, $\mathcal{V} = \{S_1 || \dots\}$, and use this symbol consistently for the actual voting options throughout the document.

Related Issues

EX-384 (planned), EX-400 (planned)

Created

04/09/2025

Status

Planned

EX-384: Domain Correctness Information**Description**

The domain of the correctness information is only implicitly defined in the System Specification 1.5.0. Section 3.5.2, which describes the correctness information, does not specify a domain. Later in the document, wherever the `pTable` is used within an algorithm, the domain is defined as \mathcal{T}_1^{50} . This appears to be correct, as only a single character, along with the pipe symbol, may be encoded at the beginning of the value. However, in the implementation, the correctness information is validated either as a \mathcal{T}_1^{50} element or as a $\mathcal{T}_1^{50} | \mathcal{T}_1^{50}$ element; a special domain, similar to that of the actual voting option.

Recommendation

Explicitly define the correct domain of the correctness information.

Related Issues

EX-383 (planned), EX-400 (planned)

Created

04/09/2025

Status

Planned

EX-385: Order of Primes Mapping Table**Description**

In the System Specification 1.5.1, Section 3.5.3, it is written that “*the primes mapping table must sequentially organize voting options as they appear in the election event configuration*”. The election event configuration is an XML file which is not further specified. The sentence implies that in whatever order the voting options “appear” in the configuration file, the primes mapping table adopts it. However, the sentence is followed by a detailed description of how the voting options are ordered: “*We first order the voting options by `votePosition` and `electionGroupPosition`. Then, we sort...*”. It remains unclear what the terms “`votePosition`”, “`electionGroupPosition`”, etc., refer to, and whether this order must be established during the creation of the primes mapping table or verified when creating it.

As we couldn’t find a code fragment in the implementation that establishes or verifies the described order, we must assume that the primes mapping table is simply created based on how the voting options “appear” in the configuration file. Therefore, the newly added description is more confusing than helpful, and it doesn’t add any value to the specification.

Recommendation

Provide a detailed specification of the complete election event configuration. If the order of voting options within the primes mapping table is essential, include a clearly defined algorithm that establishes the required order.

Created

04/09/2025

Status

Clarified

EX-386: Specification Alg. 3.11 GetHashElectionEventContext**Description**

In earlier versions, the value $h_{\text{pTable},j}$ on Line 2 was defined as a tuple containing two elements: the first being a tuple of p , q , and g , and the second being the `pTable`. In Version 1.5.0 of the System Specification, the (p, q, g) -tuple has been removed, and the $h_{\text{pTable},j}$ now consists solely of the `pTable`. However, the tuple braces were not removed accordingly, resulting in an unnecessarily nested structure (which in fact is not implemented).

Recommendation

Remove the unnecessary braces. Since $h_{\text{pTable},j}$ is now identical to the `pTable`, and the `pTable` is already defined in the context, the entire Line 2 is redundant and should be removed. Instead, the `pTable` should be placed directly into the $h_{\text{vcs},j}$ tuple in place of $h_{\text{pTable},j}$ (which would also improve consistency with the actual implementation).

Created

04/09/2025

Status

Planned

EX-387: Implementation Alg. 3.11 GetHashElectionEventContext**Description**

To improve the auditability and the alignment with the specification, it would be good to implement the algorithm `GetHashElectionEventContext` as specified, and not spread the implementation over multiple context classes. The comment hasn't been updated regarding DoI.

Recommendation

Either implement `GetHashElectionEventContext` as specified, or remove the algorithm completely and instead specify the data and structure of an election event (context) in detail. Computing the hash value is then straightforward using the `RecursiveHash` algorithm.

Created

04/09/2025

Status

Planned

EX-388: Hashing the pTable in GetHashElectionEventContext and GetHashContext**Description**

In both Alg. 3.11 `GetHashElectionEventContext` and Alg. 3.12 `GetHashContext`, the `pTable` is included in the hash computation. However, `GetHashElectionEventContext` applies `RecursiveHash` directly to the `pTable`, while `GetHashContext` first constructs an explicit structure from the individual `pTable` values before hashing.

Recommendation

Either provide a justification for why the `pTable` must be hashed differently in the two algorithms, or align the algorithms by applying a consistent hashing method; preferably as in `GetHashElectionEventContext`.

Created

04/09/2025

Status

Planned

EX-389: Extract Algorithms and Functions

Description

In the System Specification 1.5.0, a new concept based on so called **Extract** algorithms and functions has been introduced. We see no added value in these new algorithms and functions and have identified several conceptual issues. In our view, it would have been sufficient—and preferable—to specify well-defined parameters and data structures instead of introducing this new concept (see recommendation).

All **Extract** functions are just “assumed to exist”, which is problematic in the context of a formal specification. Functions should not be assumed but explicitly specified. The input and output values of these functions are left unspecified, and we question whether they really are proper functions in the mathematical sense. It appears that these “functions” are not self-contained but instead depend on external resources or state, binding them to a broader context. As such, they are methods rather than pure functions and deviate from the formal notion of a function.

The **Extract** algorithms are not algorithms in a mathematical sense, as they rely on external state and resources indirectly accessed via the **Extract** function. This approach contradicts earlier discussions and agreements regarding the separation of context, state, and algorithms.

The algorithms are poorly specified and difficult to interpret or review. For instance, in Alg. 3.13 **ExtractElectionEvent**, three local variables are assigned the return values of **Extract** functions, yet these variables are never used explicitly within the algorithm. Instead, they are only used implicitly as context in other algorithms, which introduces ambiguity. Moreover, it is unclear why values such as $\mathbf{EL}_{\mathbf{pk}}$ and \mathbf{pk}_{CCR} need to be extracted, given that they are directly available in the context of other algorithms (e.g., Alg. 5.5 **VerifyBallotCCR** or Alg. 5.7 **DecryptPCC**). Additionally, the algorithm **ExtractVerificationCardSet** is invoked inside a for-loop without any arguments:

```
5: for i \in [0,N_bb) do
6:   evcs_i <- ExtractVerificationCardSet()
7: end for
```

Not even the loop variable i is passed to **ExtractVerificationCardSet**, which raises serious questions about the correctness and clarity of the algorithm. In fact, all data used by **ExtractVerificationCardSet** are passed as context variables—despite the statement in Section 1.5.1 that “*context variables remain invariant across multiple algorithm invocations*”. This contradicts the intended use of context and undermines the consistency of the specification.

In **GetHashExtractedElectionEvent**, the first four lines are not only confusing but also not required at all.

```
1: for i \in [0,N_bb) do
2:   h_evcs,i <- (h, vcs, L_pCC, L_lVCC)
3: end for
4: h_evcs <- (h_evcs,0, ..., h_evcs,N_bb-1)
5: h <- (hContext, (p,q,g), ee, h_evcs)
```

```
6: d <- Base64Encode(RecursiveHash(h))
```

In Line 2, the same values are repeatedly assigned to the indexed $h_{\text{evcs},i}$ variable, which is clearly incorrect. Moreover, the first four lines of the algorithm are entirely redundant, as the resulting h_{evcs} is identical to the evcs already available as a context variable. Thus, only Line 5 is necessary, using evcs directly in place of h_{evcs} . Additionally, the resulting hash value produced by this algorithm includes several values more than once (such as each pTable appearing twice, and values like ee and (p, q, g) appearing multiple times). We do not see any purpose or added value in this repetition, but consider it the result of the lack of a well-defined specification for the parameters and data structures involved.

As we understand it, the purpose of all **Extract** functions and algorithms is simply to gather data in order to compute a final hash over the public state of a given control component. Achieving this goal does not require separate extractors; instead, it would be sufficient to define clear parameters and data structures. For instance, an election event could be specified as a tuple $(id, \text{eeAlias}, \text{eeDesc}, (p, q, g), \dots)$ from which the hash can then be directly and unambiguously computed using the **RecursiveHash** algorithm.

Recommendation

We recommend removing all newly introduced **Extract** functions and algorithms, and instead providing well-defined parameters and data structures to serve the same purpose in a clearer and more consistent manner.

Created

04/09/2025

Status

Planned

EX-390: Precondition on ExtractVerificationCards

Description

In the preceding paragraph to Alg. 3.16 **ExtractVerificationCards**, it is written that “*as a precondition, the ExtractVerificationCards algorithm can only be invoked after the election event period ended*”. Why is that? We agree that calling the algorithm during the voting phase or even before the voting phase might not make much sense. But it also doesn’t break anything. So, by stating this precondition, it also needs to be checked, which, however, isn’t trivial as it requires a greater context.

Created

04/09/2025

Status

Clarified

EX-391: Specification Alg. 4.6 CombineEncLongCodeShares

Description

The exponentiated, encrypted, hashed partial Choice Return Codes $\mathbf{C}_{\text{expPCC}}$ is named

“matrix” but defined as a vector of vectors (in the same way as π_{expPCC}). By inspecting the code, C_{expPCC} is implemented as a matrix and π_{expPCC} as a vector of vectors.

Recommendation

Change the definition of C_{expPCC} in the specification to $C_{\text{expPCC}} \in (\mathbb{G}_q^{n+1})^{N_E \times 4}$. Dito for C_{expCK} .

Created

04/09/2025

Status

Planned

EX-392: System Specification Section 6.1 MixOnline

Description

In the System Specification 1.5.0, it is written that “*each control component ensures that its view of the initial ciphertexts is consistent with the other control components before initiating the mixing process*” (see Section 6.1.3). It is further written that if their views are not consistent, then the process halts and the Dispute Resolver run by a third-party will resolve the discrepancies. As this is a really important and critical step of the MixOnline protocol involving a complete communication round among the control components and a potential process halt, it should be depicted in the sequence diagram Fig. 14.

Recommendation

Update the sequence diagram Fig. 14 in the System Specification

Created

04/09/2025

Status

Clarified

EX-393: Specification Alg. 6.4 CheckExtractedElectionEventConsistency

Description

Calling an algorithm in a loop without any input arguments and with “invariant” context is conceptually flawed and difficult to comprehend.

```
1: for j \in [1,4] do
2:   h_eeej <- GetHashExtractedElectionEvent()
3: end for
```

Please improve the readability and non-ambiguity in the algorithm specification.

Related Issues

EX-317 (planned), EX-318 (planned), EX-319 (planned), EX-3290 (planned)

Created

04/09/2025

Status

Clarified

EX-394: Specification Alg. 6.5 CheckVoteConsistency**Description**

On Line 3 within in the nested loop, the indices on the right-hand side of the assignment are missing.

```

1: for j \in [1,4] do
2:   for i \in [0,N_S] do
3:     h_E1_j,i <- (\gamma_1, \phi_1,0, ..., \phi_1, \delta_1)

```

Created

04/09/2025

Status

Clarified

EX-395: Specification Alg. 6.6 CheckVoteConfirmationConsistency**Description**

The new algorithm Alg. 6.6 CheckVoteConfirmationConsistency has a number of problems:

- The name of the algorithm is confusing, as the algorithm doesn't do any check (in contrast to CheckExtractedElectionEventConsistency and CheckVoteConsistency). It just builds the union of all confirmed votes across the control components.
- The extracted election event `eee` is far too much as context. Only `ee` and L_{1VCC} are required.
- Line 4 is unnecessarily complicated and hence very difficult to read, understand, and implement correctly.
- Line 5 is completely useless. Especially since most of the values of $evc_{j,i}$ are already used on line 4.
- Calling `ConfirmVoteAgreement` with only the $h1VCC$ values as input, but not also with vc_{id} and vcs , is confusing.

Recommendation

Change the name of the algorithm and specify the context parameters more accurately. To improve the readability of the algorithm and the alignment with a possible implementation (the algorithm as specified now will hardly be implemented like this) change the algorithm to something like:

```

1: rcv <- ()
2: for j \in [1, 4] do
3:   for i \in [0, N_S] do
4:     (vc_id, vcs, E1, h1VCCs) <- evc_j,i
5:     if (vc_id, \cdot, \cdot) \not_in rcv AND h1VCCs \not_eq () then
6:       if ConfirmVoteAgreement(vc_id, vcs, h1VCCs) then

```

```
7:         rcv <- rcv || (vc_id, vcs, hlvCCs)
8: ...
```

(Note: Line 4 and 5 are very close to the current implementation)

Created

04/09/2025

Status

Planned

EX-396: System Specification Section 7 Channel Security

Description

Up to System Specification 1.5.0, the contents and order of values included in the authenticated messages were explicitly defined in Tables 15, 16, and 17. This information, together with the well-specified context data, is not only essential for creating and verifying the message signature, but also very helpful for reading and understanding the System Specification. Unfortunately, this very helpful information has been removed in Version 1.5.0 and replaced by the name of a JSON schema specifying the message payload. A new table (Table 18) has been added, listing all payload schemas along with their referenced JSON schemas. This change introduces two major problems:

1. The specification is no longer self-contained. The JSON schemas are not included in the document and must instead be retrieved from the code base.
2. JSON schemas do not and cannot specify the order of elements. As a result, essential information needed to compute and verify digital signatures is missing. Moreover, it remains unspecified how multiple values are to be hashed; whether as a tuple, as individually hashed and concatenated values, or otherwise.

Recommendation

Add the message content, as it used to be defined (a tuple of values) again to the message tables. It is fine, if the JSON schemas are listed as well, but they don't suffice. However, we also think that JSON schemas are an implementation detail and shouldn't be part of a specification.

Related Issues

EX-271 (delivered), EX-380 (planned)

Created

04/09/2025

Status

Planned

EX-397: Specification Section 7.2 Digital Signatures for File Interfaces (XML)

Description

A new Section 7.2 about the generation and verification of XML signatures has been

added to the System Specification 1.5.0. It is written that “*given the complexity of these XML structures, providing detailed pseudocode describing their structure would be impractical. Instead, we rely on the standardized XML signature syntax and verification methods as defined in [1], which are supported by most programming languages*”. Indeed, by inspecting the code base, the generation and verification of the XML signatures is delegated to a library (`javax.xml.crypto.dsig`). Nevertheless, two new algorithms, Alg. 7.1 `GenXMLSignature` and Alg. 7.2 `VerifyXMLSignature`, have been added to the specification. Based on our analysis, both algorithms are incorrect and refer to a number of functions which are not further specified. Hence, the two new algorithms don’t provide any added value and contradict the statement that they follow the standard. By using a standard, there is no need to re-specify its details in pseudocode (in the same way as no pseudocode is provided for AES or Argon2).

Recommendation

We recommend removing the two newly introduced algorithms. It is perfectly fine to describe the standard on a high level and to outline the steps a signing library must perform. However, providing algorithms that are meant to be implemented exactly as specified is redundant and potentially misleading; that is what the standard is for.

Related Issues

EX-273 (delivered)

Created

04/09/2025

Status

Clarified

EX-398: Specification Alg. 7.3 GenStreamCiphertext

Description

In the System Specification 1.5.0, a new section has been added to specify the symmetric encryption and decryption of data exchanged between offline and online components. Surprisingly, the algorithm Alg. 7.3 `GenStreamCiphertext` specifies exactly what the preceding paragraph states should not be done: “*As explained in the crypto-primitives specification, it is critical that nonces should not be reused*”. However, within the while loop on Line 6, the same nonce is reused again and again. In fact, Alg. 7.3 `GenStreamCiphertext` specifies the usage of AES (the symmetric cipher used in the algorithm `AuthenticatedEncryption`) in a manner similar to ECB mode. As the block size used by `readNextBlock` is not specified, it could be as small as 128 bits, but even larger block sizes will leak information about the encrypted plaintext. Using AES in ECB mode is known to be a problem for more than thirty years (in fact, it has been known that ECB leaks information even before AES existed). As each block is encrypted individually using AES-GCM under the hood, nonce re-use affects not only confidentiality but also message integrity, as clearly described by Aleksander Essex in his examination report from May 2025 [Essex25]. This raises concerns regarding the internal cryptographic expertise and the internal review process.

```

1: (derivedKey, salt) <- GenArgon2id(toBytes(password))
2: nonce <- RandomBytes(12)
3: C <- <>
4: while ¬endOfData(P) do
5:   block <- readNextBlock(P)
6:   encryptedBlock <- AuthenticatedEncryption(derivedKey, nonce,
       block, associated)
7:   C <- C || encryptedBlock
8: end while

```

Recommendation

- Strengthen the internal review process. Such publications are highly problematic and risk undermining public trust in Swiss Post and general in e-voting in Switzerland.
- Since the implementation does not follow the specification, remove the newly specified algorithms for streamable encryption and decryption. Instead, write that, given the large size of the datasets, the used encryption library should support streamable input in order to avoid loading the entire dataset into memory at once. This approach is also consistent with the current implementation.

Created

04/09/2025

Status

Delivered (fixed in Release 1.5.2)

EX-399: Dispute Resolver

Description

The System Specification 1.5.0 elaborates in detail on the *Dispute Resolver*. In Section 2.10 the Dispute Resolver is defined as an “*independent party*” running “*in a controlled, offline environment on cantonal premises*”. From this, we conclude that the Dispute Resolver is an expensive party in operation: not only is it laborious and inefficient to really use the Dispute Resolver, but even ensuring its availability as an independent, offline component on cantonal premises is costly. Moreover, the introduction of this additional party complicates the analysis of the protocol. If the protocol allows for disputes among the control components that can only be resolved by an independent party, then we think such a Dispute Resolver is justified. However, if no such disputes can arise, then the Dispute Resolver is unnecessary.

If the Dispute Resolver detects that the control components disagree on the list of sent ballots (that is, the list of ballots sent by voters including confirmed and non-confirmed ballots), then the Dispute Resolver cannot do anything. The process is stopped and a forensic investigation must be initiated, since at least one control component acts maliciously. This is due to the fact that “*the SendVote protocol establishes an explicit agreement on the election event context and the encrypted votes*”.

If the control components don’t agree on the list of confirmed ballots, then the Dispute Resolver computes the union of the confirmed ballots among the control components.

However, only confirmations are taken into consideration if a control component can provide the opening of a commitment in the L_{1VCC} list. To provide such an opening, a control component must be in possession of a share from each of the other control component. This proves that each control component must have received the correct confirmation code from the voter and consequently the vote must be regarded as confirmed from the voter's perspective. If, after this reconciliation, a control component still rejects the updated list of confirmed votes, then we have a similar situation as above—a malicious control component—and a forensic investigation must be initiated.

Because the Dispute Resolver runs an entirely deterministic protocol (neither the canton, nor Swiss Post, nor the Dispute Resolver itself makes judgments, only facts are combined), the protocol could be run just as well amongst the control components themselves. That is, during the `MixOnline` protocol, while exchanging the hash of the encrypted and confirmed votes, the control components could also provide evidence for their views (namely the `h1VCC` values which result in an opening in the L_{1VCC} list). Each control component could then independently build the union of confirmed votes. If afterwards a control component rejects a shuffle proof in the `MixOnline` phase, a control component must be acting maliciously and a forensic investigation must be initiated. The outcome would therefore be identical to that achieved with a Dispute Resolver, but reached in a way that is simpler, cheaper, and far less complex.

We are aware that the Dispute Resolver is listed in the Massnahmenkatalog. As far as we remember, this is due to an earlier protocol version in which real disputes between the control components could have occurred. Thanks to the introduction of the long vote cast return codes allow list (L_{1VCC}), this is no longer the case, and we propose removing this measure from the list.

Recommendation

Remove the newly introduced Dispute Resolver, as the protocol doesn't require the involvement of such a party. Instead, slightly enhance the consistency round amongst the control components in the `MixOnline` protocol.

Related Issues

EX-374 (in progress)

Created

04/09/2025

Status

Clarified

EX-400: Implementation PrimesMappingTableEntry Domain Checks

Description

The confusion with the domain of the actual voting option (see Issue EX-383) is clearly reflected in the code.

```
@param actualVotingOption v<sub>i</sub>, the actual voting option. Must be non-null. It must not be blank and its length must be at most {@value ch.post.it.evoting.evotinglibraries.domain.VotingOptionsConstants#MAXIMUM_ACTUAL_VOTING_OPTION_LENGTH}.
```

The constant `MAXIMUM_ACTUAL_VOTING_OPTION_LENGTH` is set to 50, which is incorrect, as an actual voting option can have a length of up to 152 characters. On the other hand, the correctness information, which should be from the domain \mathcal{T}_1^{50} , has the following comment:

```
@param correctnessInformation i, the correctness information related to the voting option. Must be non-null and a valid combination of xml xs:token.
```

Finally, the validator for `xs:token` enforces a maximal length of `MAXIMUM_ACTUAL_VOTING_OPTION_LENGTH`. This is highly misleading, since an `xs:token` in general, and the correctness information in particular, are not related to the actual voting option at all.

We don't claim that the validation is incorrect. We only say that the terminology and the comments are very confusing.

Recommendation

Define appropriate domains for the actual voting options and the correctness information, and thoroughly align the code.

Related Issues

EX-383 (planned), EX-384 (planned)

Created

04/09/2025

Status

Planned

EX-401: Implementation Diverges from Specification

Description

Following is a list of a few exemplary examples showing that the implementation does not strictly follow the specification. The examples don't criticize the implementation but indicate that the specified concepts are difficult to implement, resulting in weak alignment between implementation and specification.

- `ExtractVerificationCardSet` (confusion context/input parameters):
 - Specified: `h ← GetHashContext()`
 - Implemented: `h = getHashContextAlgorithm.getHashContext(p_q_g, ee, vcs, pTable, EL_pk, pk_CCR);`
- `GetHashExtractedElectionEvent` (irrelevant and useless pseudocode):
 - Specified is an algorithm of six lines including a for-loop;
 - Implemented is a one-liner:
`return base64.base64Encode(hash.recursiveHash(extractedElectionEvent));`
- `ExtractVerificationCards` (lack of well-defined parameters and data structures):
 - Specified: Loop over `vcid`
 - Implemented: Loop over votes (complex object)

- Specified: $\text{vcs} \leftarrow \text{ExtractVerificationCardSetId}()$
Implemented: `vc_id = sentVote.contextIds().verificationCardId();`
- Specified: $\text{E1} \leftarrow \text{ExtractEncryptedVote}(\text{vc_id})$
Implemented: `E1 = sentVote.encryptedVote();`

Recommendation

Improve the specification to achieve closer alignment with an implementation.

Created

04/09/2025

Status

Clarified

EX-402: Alg. 4.7 [...] and Alg. 4.8 [...]

Description

In Alg. 4.7 `VerifyEncryptedPCCExponentiationProofs` \mathbf{c}_{pcc} is specified as an element of $(\mathbb{G}_q^{n+1})^{N_E}$. Implemented, on the other hand, is the following check:

```
c_pcc.getElementSize() == n
```

Although it is correct, specifying a list to be of size $n + 1$ and then checking for size n is confusing and error-prone. We recommend clarifying the specification with a more precise domain definition, for example: $\mathbf{c}_{\text{pcc}} \in (\mathbb{G}_q \times \mathbb{G}_q^n)^{N_E}$. The same holds for $\mathbf{c}_{\text{expPCC},j}$.

In Alg. 4.7 `VerifyEncryptedPCCExponentiationProofs` and Alg. 4.8 `VerifyEncryptedCKExponentiationProofs` the logical reduction

```
.reduce(Boolean::logicalAnd).orElse(Boolean.FALSE)
```

is incorrect. The neutral element for an AND composition is *true*, not *false*. Therefore, it should be:

```
.reduce(Boolean::logicalAnd).orElse(Boolean.TRUE)
```

Created

04/09/2025

Status

Clarified

EX-403: Implementation Alg. 6.4 [...] and Alg. 6.5 [...]

Description

Both algorithms Alg. 6.4 `CheckExtractedElectionEventConsistency` and Alg. 6.5 `CheckVoteConsistency` have specified exactly the same final check. In both cases, four Base64-encoded strings are compared and if they are equal, \top is returned, otherwise \perp :

```
1: # CheckExtractedElectionEventConsistency
2: return h_eee1 = h_eee2 = h_eee3 = h_eee4 ? \top : \bot
```

```

1: # CheckVoteConsistency
2: return d_1 = d_2 = d_3 = d_4 ? \top : \bot

```

However, the final check is implemented completely differently in the two algorithms:

```

1: # CheckExtractedElectionEventConsistency
2: final String h_eee_1 = h_eee_vector.get(0);
3: final String h_eee_2 = h_eee_vector.get(1);
4: final String h_eee_3 = h_eee_vector.get(2);
5: final String h_eee_4 = h_eee_vector.get(3);
6:
7: // h_eee_1 = h_eee_2 = h_eee_3 = h_eee_4
8: return h_eee_1.equals(h_eee_2) && h_eee_2.equals(h_eee_3)
    && h_eee_3.equals(h_eee_4);

1: # CheckVoteConsistency
2: // d_1 = d_2 = d_3 = d_4
3: return [...].distinct().count() == 1;

```

Although both implementations are correct, different implementations of the same patterns reduce maintainability and make the review process unnecessarily complex.

Recommendation

We recommend streamlining the implementation by consistently applying the same patterns.

Created

04/09/2025

Status

Panned

2.3. Voting-Card-Print-Service and PDF-Verification-Service

The two new components added recently to the code base have already been examined by other experts [OH25]. Therefore, we only reviewed them superficially, i.e., without looking at the details of the implemented code. We added two issues EX-410 and EX-411 to the Jira platform with a few general remarks.

EX-410: Service name misleading: PDF Verification Service [...]

Description

The service name is misleading, as it does not verify PDFs but rather performs authenticated decryption of encrypted ZIPs and signature verification of their contents.

The current name **PDF Verification Service** implies that the primary role of the service is to verify PDFs. However, the actual functionality does in fact other things—and differs from verification:

- The service accepts symmetrically AEAD-encrypted ZIP files containing PDFs and digital signatures.

- It first performs AEAD decryption of the ZIP archive, which ensures authenticity and integrity of the encrypted container.
- It then extracts the PDFs and verifies the attached digital signatures.
- Only if both steps succeed the service outputs the plaintext PDFs, otherwise, the process ends with an exception (intended).

Verification vs. Decryption semantics:

In cryptographic and security terminology and especially in our e-voting context, a verification process yields a boolean result (accepting/rejecting the process), but does never produce any other output, hence, no decrypted plaintexts. However, this service does produce decrypted PDFs when authenticity and integrity checks out—the characteristic of an authenticated decryption process and not of a verification.

PDF verification is misleading:

Moreover, the service does not verify the content of the PDF documents themselves. It verifies only the cryptographic authenticity of the container and signatures. Therefore, the term “*PDF Verification*” may wrongly suggest that the PDF’s structure or contents are validated as well, which is not the case.

Recommendation

To better describe the true purpose of the service, a clearer name might be:

- Authenticated PDF Decryption Service
- Decryption and Signature Verification Service

This would avoid the confusion that the PDFs themselves were being “verified”. But it would highlight the core functionality of the service: authenticated decryption with signature checking.

Created

17/09/2025

Status

Clarified

EX-411: PDF Verification Service: Overengineered implementation [...]

Description

A service that should be a simple decryption script has been built on a huge tech stack, making actually hard to truly audit. Moreover, this particular implementation exposes the complete process to unnecessary supply-chain attack risks. This is particularly dangerous as the service operates at the point where all voting cards are revealed, thus the point of maximal trust.

The service’s is conceptually simple (as described in EX-410):

- Perform AEAD decryption of a ZIP archive.
- Verify the attached digital signatures.
- Release the decrypted PDFs only if authenticity and integrity checks succeed.

But Swiss Post decided to provide an implementation using a very large tech stack (Spring Boot backend, Angular frontend, Electron deployment).

Why we consider it a critical issue?

- *Auditability*: Independent reviewers cannot easily confirm the service does only what it claims, because of unnecessary complexity.
- *Trust minimization*: Best practice is to keep the trusted computing base small. Here, every extra dependency increases risk.
- *Supply-chain attacks*: Large dependency trees and toolchains create real attack surfaces, as shown by recent compromises such as the *3CX Desktop App* compromise (2023), the *XZ Utils* backdoor (2024), the *npm Chalk/Debug* incident (2025), and the *Shai-Hulud* supply chain attack (2025).
- *Most vulnerable moment* This service unveils all voting cards. A compromise here allows to breach voter privacy or in a more complex scenario even correctness, by swapping ballot contents (this indeed is a complex attack, but it cannot be ruled out). Swiss Post mentions “hardened computers”, but without clear and verifiable detail at this point, this mitigation must be considered insufficient.

Recommendation

A simple, auditable script would suffice for the required functionality. The current “cathedral” design clearly adds many risks without true benefit.

- Switch to a minimal reference implementation (small script/CLI) with least crypto and file I/O dependencies.
- Keep ensuring reproducible builds and signed releases.
- Document clearly what the exact requirements of the “*hardened computers*” are at this point in order to minimize the attack surface in practice.

Reducing complexity at this critical step makes the system maintainable, auditable, harder to attack, and safer for both privacy and correctness.

Created

17/09/2025

Status

Clarified

A. Changelogs

A.1. Crypto-Primitives

Component	Version	Changes listed in the component's CHANGELOG.md file
crypto-primitives	1.5.0	<p>Release 1.5.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community.</p> <ul style="list-style-type: none">• Allow edge cases in several algorithms, and so adhere to the Principle of Generality. This affects the pseudocode and implementation of the following algorithms: <code>CutToBitLength</code>, <code>ByteArrayToInteger</code>, <code>ByteLength</code>, <code>ByteArrayToString</code>, <code>Truncate</code>, <code>GenRandomVector</code>, <code>GenRandomString</code>, <code>GenUniqueDecimalStrings</code>, <code>GenCiphertextSymmetric</code>, <code>GetPlaintextSymmetric</code>, <code>GenPermutation</code>.• Systematically add modulo operator in pseudocode. This affects the pseudocode of the following algorithms: <code>StarMap</code>, <code>GetShuffleArgument</code>, <code>VerifyShuffleArgument</code>, <code>GetMultiExponentiationArgument</code>, <code>VerifyMultiExponentiationArgument</code>, <code>GetProductArgument</code>, <code>GetHadamardArgument</code>, <code>VerifyHadamardArgument</code>, <code>GetZeroArgument</code>, <code>VerifyZeroArgument</code>, <code>ComputeDVector</code>, <code>VerifySingleValueProductArgument</code>, <code>ComputePhiDecryption</code>, <code>GenDecryptionProof</code>, <code>VerifyDecryption</code>, <code>VerifyExponentiation</code>, <code>ComputePhiPlaintextEquality</code>, <code>GenPlaintextEqualityProof</code>, <code>VerifyPlaintextEquality</code> (reported in GitLab Issues 52 and 20).• Improve variable naming in algorithm <code>IntegerToFixedLengthByteArray</code>.• Simplify <code>IntegerToByteArray</code> and <code>IntegerToFixedLengthByteArray</code>.• Improve variable naming in algorithms <code>KDF</code> and <code>KDFToZq</code>.• Add requirement in <code>KDFToZq</code>.• Improve the description of how to generate a signing key and certificate and ensure the generation of certificates complies with RFC 5280 (reported in GitLab Issue 17).• Improve the pseudocode of the algorithm <code>GenKeysAndCert</code>.• Improve the description of the output of algorithms <code>GenSignature</code> and <code>VerifySignature</code>.• Shorten the description of <code>GenKeyPair</code> by removing textbook-style details.• Improve variable naming in the algorithm <code>GenKeyPair</code>.• Improve variable naming in the algorithm <code>GetCiphertextVectorExponentiation</code>.• <code>GenShuffle</code> small alignment to specification.

Component	Version	Changes listed in the component's CHANGELOG.md file
crypto-primitives	1.5.0	<ul style="list-style-type: none"> • Extend test cases for algorithm VerifyDecryption (reported in GitLab Issue 23). • Add hash values to test data for VerifyShuffleArgument, VerifyMultiExponentiationArgument, VerifyHadamardArgument (reported in GitLab Issue 21). • Replace base16 by base64 in JsonData and json test files. • Ensure usage of $g = 4$ in test vectors. • Use new crypto-primitives type ImmutableList instead of List for collections simply holding immutable data. • Replace the usage of List<String> auxiliaryInformation by the new type AuxiliaryInformation. • Replace the usage of byte[] by the new type ImmutableByteArray. • Ensure immutability for map and set objects. • Use ImmutableArray whenever we use vectors. • Change records with multiple arguments of same type to class and add builder. • Include reasoning for using both SHA3-256 and SHA-256. • Clarification in Table 2 about group parameters used for the testing-only security level. • Clarification in Table 3 about nonce size and key size of the symmetric algorithm for authenticated encryption and decryption. • Clarification in table 8 about the variables used for the Argon2 profiles. • Minor corrections and clarifications. • Minor improvements. • Update dependencies and third-party libraries.
crypto-primitives	1.5.1	<p>Release 1.5.1 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...] The following functionalities and improvements are included in Release 1.5.1:</p> <ul style="list-style-type: none"> • Improve byte notation consistency in algorithm CutToBitLength (feedback from Olivier Pereira). • Correct the statement about the bit length of 0 (reported in GitLab Issue 24). • Update dependencies and third-party libraries.
crypto-primitives	1.5.2	<p>Release 1.5.2 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Isolated improvements to build reproducibility. • Update dependencies and third-party libraries.
crypto-primitives	1.5.2.1	<p>Release 1.5.2.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Update dependencies and third-party libraries.

Component	Version	Changes listed in the component's CHANGELOG.md file
crypto-primitives-ts	1.5.0	<p>Release 1.5.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...] The following functionalities and improvements are included in release 1.5.0:</p> <ul style="list-style-type: none"> • Internalize the dependencies jsSHA and utf8. Thereby, the crypto-primitives-ts repository no longer has any external dependencies (in line with measure A.10 of the Federal Chancellery's catalogue of measures). • Ensure immutability for map and set objects. • Minor improvements. • Updated development dependencies.
crypto-primitives-ts	1.5.1	<p>Release 1.5.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Updated dependencies and third-party libraries.
crypto-primitives-ts	1.5.2	<p>Release 1.5.2 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Isolated improvements to build reproducibility. • Updated dependencies and third-party libraries.
crypto-primitives-ts	1.5.2.1	<p>Release 1.5.2.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Updated dependencies and third-party libraries.

A.2. Data Integration System

Component	Version	Changes listed in the component's CHANGELOG.md file
data-integration-service	2.9.0	<p>The Data Integration Service 2.9.0 contains the following changes:</p> <ul style="list-style-type: none"> • Added eCH-version 4.2 support. • Improved input validation. [...] • Updated the data integration service to evoting-config version 7. [...] • Included the new XML signature for the evoting-config. • General code quality improvements and minor bug fixes. • Updated dependencies and third-party libraries.
data-integration-service	2.9.1	<p>Release 2.9.1 incorporates the following changes:</p> <ul style="list-style-type: none"> • Improved a protection against XML external entity attacks. • Added sanitization of markdown links • Fixed a limitation when handling large number of municipalities. • Fixed a problem when treating custom blank answers. • Enforced a consistency check in the counting circle identifier to counting circle name mapping. • Minor code improvements • Updated dependencies and third-party libraries.
data-integration-service	2.9.2	<p>Release 2.9.2 incorporates the following changes:</p> <ul style="list-style-type: none"> • Improved the sanitization of the non-anonymized configuration file. • Added additional display candidate line styles. • Fixed a bug with overlapping vote and election group positions. • Ensured correct treatment of the country code "NO". • Added a default list order of precedence for the empty list. • Fixed a bug with logging signature exceptions. • Minor code improvements • Updated dependencies and third-party libraries.
data-integration-service	2.9.2.1	<p>Release 2.9.2.1 incorporates the following changes:</p> <ul style="list-style-type: none"> • Added a validation that candidate accumulation does not exceed the number of mandates. • Parametrized whether the incumbent text is displayed on the display candidate line. • Added a testAuthorizationSplitter to split the output files for test voter by authorization. • Minor code improvements. • Updated dependencies and third-party libraries.
data-integration-service	2.9.2.2	<p>Release 2.9.2.2 is a minor maintenance patch correcting the following issue:</p> <ul style="list-style-type: none"> • Fixes a problem with inconsistent numberings in election groups and popular votes.

A.3. E-Voting

Component	Version	Changes listed in the component's CHANGELOG.md file
e-voting	1.5.0	<p>Release 1.5.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. Release 1.5.0 includes the following functionalities and improvements:</p> <ul style="list-style-type: none"> • Include the verification of the zero-knowledge proofs generated by the control components during the configuration phase. Previously, these verifications were part of the verifier. • Add the verification of the zero-knowledge proofs in the algorithm CombineEncLongCodeShares. • Remove the export of the setup dataset in the Secure Data Manager. • Replace the remaining JavaScript code with TypeScript in the voting client (renaming it from voting-client-js to voting-client) and remove the lodash dependency (in line with the measure A.10 to reduce third-party dependencies). • Use GetHashExtractedElectionEvent in PartialDecryptPCC and DecryptPCC. • Implement the dispute resolver to allow resolving inconsistent view of confirmed votes across the control components (in line with the measure A.21 of the Federal Chancellery's catalogue of measures). • Implement new extraction algorithms: ExtractElectionEvent, ExtractVerificationCardSet, and ExtractVerificationCards. • Implement new dispute resolver algorithms: CheckExtractedElectionEventConsistency, CheckVoteConsistency, CheckVoteConfirmationConsistency • Implement new algorithm: UpdateConfirmedVotingCards. • Enhance the voter portal based on expert feedback on e-voting usability and to promote the use of verification features (in line with the measure A.19 of the Federal Chancellery's catalogue of measures). • Align the portal layout and terminology with the printed voting card. • Separate the verification of return codes from the entry of the ballot casting key. • Introduce a dedicated finalization page to confirm that the voting process is complete. • Improve the UX elements for popular votes and elections. • Remove the generation of the evoting-decryption and eCH-0110 XML files. • Use the new algorithms GenXMLSignature and VerifyXMLSignature for all XML interfaces.

Component	Version	Changes listed in the component's CHANGELOG.md file
e-voting	1.5.0	<ul style="list-style-type: none"> • Streamline the import of an election event into the Secure Data manager and remove the OrientDB with a more standard relational SQLite database. • Add a detailed per ballot box display of the election event result (for both popular votes and elections). • Improve the handling of various edge cases based on the input of previous election events. • Improve validation of election events when accessing the voter portal. • Improve checks for device compatibility. • Implement more robust handling of repeated vote confirmation requests. • Allow the handling of multiple tenants on the same platform (multi-tenancy), thereby reducing the need of deploying multiple redundant platforms and streamlining the infrastructure. • Upload the voter portal configuration from the Secure Data manager. • Align the different API accordingly for multi-tenancy. • Refactor the management of databases and allow the full transition from Oracle databases to PostgreSQL. • Improve robustness when handling very large election events. • Various small improvements and bug fixes. • Updated dependencies and third-party libraries.
e-voting	1.5.1	<p>Release 1.5.1 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...] Release 1.5.1 includes the following functionalities and improvements:</p> <ul style="list-style-type: none"> • Added a consistency check of the certificates when importing them in the direct trust tool (feedback from Thomas Haines). • Changed extracted election event from context to input in CheckExtractedElectionEventConsistency (feedback from Thomas Haines). • Included an edge case for handling an empty list of confirmed votes in the dispute resolver (feedback from Thomas Haines). • Added sanitization of parameters in electron application (feedback from Orange Cyberdefense). • Improved performance by caching and saving the hashed extracted election event in the control components. • Improved translations in the voter portal. • Improved certain exceptional workflows in the voter portal. • Added tally output containing the eCH-0222.xml in the Tally SDM. • Fixed various display issues in the voter portal. • Various small improvements and bug fixes. • Updated dependencies and third-party libraries.

Component	Version	Changes listed in the component's CHANGELOG.md file
e-voting	1.5.2	<p>Release 1.5.2 incorporates the following improvements:</p> <ul style="list-style-type: none"> • Fixed various display issues in the voter portal. • Improved the voter portal message when performing a relogin after successful confirmation. • Improved deserialization of messages when downloading ballot boxes. • Improved connection retry behavior of the Secure Data Manager. • Simplified the direct trust tool application. • Allow to display an additional candidate line in the voter portal. • Improved the voter portal configuration upload in the Secure Data Manager. • Translation improvements • Improved log messages for monitoring. • Removed ngx-toaster dependency in the Secure Data Manager frontend. • Various improvements to the landing page application. • Various small improvements and bug fixes. • Updated dependencies and third-party libraries.
e-voting	1.5.2.1	<p>Release 1.5.2.1 is a minor maintenance patch and incorporates the following improvements:</p> <ul style="list-style-type: none"> • Various small bug fixes in the voter portal. • Various small improvements in the landing page application. • Added a small usability improvement in the direct trust tool by disabling the reset button for certain configurations. • Updated dependencies and third-party libraries.
e-voting	1.5.2.2	<p>Release 1.5.2.2 is a small hotfix and incorporates the following improvements:</p> <ul style="list-style-type: none"> • Minor corrections in the voter portal.

A.4. E-Voting Libraries

Component	Version	Changes listed in the component's CHANGELOG.md file
e-voting-libraries	1.5.0	<p>Release 1.5.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...] The following functionalities and improvements are included in release 1.5.0:</p> <ul style="list-style-type: none"> • Implement the algorithms GenXMLSignature and VerifyXMLSignature and use them for all XML interfaces. • Update the eCH-0222 file from Version 1.0 to Version 3.0 (eCH terminology v1.2). • Remove the XML files evoting-decrypt and eCH-0110. • Improved and streamlined the digital signature of XML files by using the XML signature syntax as defined in the system specification. • Implement new Election Event Result object structure. • Implement the new agreement algorithm GetHashExtracted-ElectionEvent. • Update the evoting-config interface to version 7.0 and the evoting-print interface to version 2.0. • Ensure correct input format to VerifyCCSchnorrProofs in the algorithm VerifyKeyGenerationSchnorrProofs [...]. • Increase the maximum supported number of selections from 120 to 150. • Extend the electoral model with election relevant information. • Implement new multi-tenancy helper classes. • Various improvements related to JSON test files and json-schema, including: [...] • Streamable Symmetric Encryption and Decryption for Dataset Security: implement the algorithms GenStreamCiphertext and GetStreamCiphertext use them where applicable. • Various improvements related to Measure A.15 of the Catalogue of Measures, namely using new immutable classes from crypto-primitives. • Minor code improvements. • Updated dependencies and third-party libraries.

Component	Version	Changes listed in the component's CHANGELOG.md file
e-voting-libraries	1.5.1	<p>Release 1.5.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Fix the incorrect order in the display of Choice Return Codes in certain types of elections. • Added normalization of the XML file in the CreateECH0222XML method. • Remove old signature field when resigning a XML document. • Fix the problem with calculating the isUnchangedBallot field in the eCH-0222 XML file. • Add additional information about elections in the election event overview. • Minor code improvements. • Updated dependencies and third-party libraries.
e-voting-libraries	1.5.2	<p>Release 1.5.2 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Fixed encoding of associated data in the GenStreamCiphertext and GetPlaintextCiphertext (reported in GitLab Issue 5). • Fixed an issue with missing logs in case of signature failures by adding an output to input stream converter. • Various improvements in the display of the ballot box results such as adding the list indenture number. • Minor code improvements. • Updated dependencies and third-party libraries.
e-voting-libraries	1.5.2.1	<p>Release 1.5.2.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Added a missing parameter in the Java Documentation of the GetHashElectionEventContext algorithm. • Isolated improvements to build reproducibility. • Updated dependencies and third-party libraries.

A.5. Voting-Card-Print-Service and PDF-Verification-Service

Component	Version	Changes listed in the component's CHANGELOG.md file
voting-card-print-service	3.3.0	<p>Release 3.3.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...]</p> <ul style="list-style-type: none"> • Addressed recommendation from the report #YWH-PGM12823-8. • Support new versions of the input files <code>configuration.xml</code> and <code>evoting-print.xml</code>. • Minor improvements to VCPS configuration. • Updated dependencies and third-party libraries.
voting-card-print-service	3.3.1	<p>Release 3.3.1 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...]</p> <ul style="list-style-type: none"> • Addressed the recommendation "<i>Remove the configurability to disable the signature and the encryption</i>" from report (Gitlab Issue 72) and feedback from Thomas Haines. • Added a validation step to enforce the correct SDK version for builds. • Added a password validation check for the file-cryptor. • New functionality for generating images for postal Datamatrix codes (DMC). • New functionality to manage multiple layouts for a single mandant. • Improved ordering of votations and elections in voting cards. • Minor layout improvements. • Minor improvements to GitLab publication. • Improvements to the build process.
voting-card-print-service	3.3.2	<p>Release 3.3.2 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...]</p> <ul style="list-style-type: none"> • Addressed the recommendation "enforce the uniqueness of MunicipalityId via the XSD". • Addressed the recommendation "sanitize VCPS.FopRenderer.FopOpts configuration value". • Addressed the recommendation "sanitize VCPS.FopRenderer.FopXconf configuration data". • Improvements to GitLab publication. • Improvements to the build process. • Updated third dependencies.

Component	Version	Changes listed in the component's CHANGELOG.md file
pdf-verification-service	1.5.0	First release of the PDF verification service.
pdf-verification-service	1.5.1	Release 1.5.1 is a minor patch containing the following changes: <ul style="list-style-type: none"> • Minor code improvements. • Updated dependencies and third-party libraries.
pdf-verification-service	1.5.2	Release 1.5.2 is a minor patch containing the following changes: <ul style="list-style-type: none"> • Improved sanitization of file names. • Improved display of the certificate fingerprints. • Minor code improvements. • Updated dependencies and third-party libraries.
pdf-verification-service	1.5.2.1	Release 1.5.2.1 is a minor maintenance patch containing the following changes: <ul style="list-style-type: none"> • Updated dependencies and third-party libraries.

A.6. Verifier

Component	Version	Changes listed in the component's CHANGELOG.md file
verifier	1.6.0	<p>Release 1.6.0 includes some feedback from the Federal Chancellery's mandated experts and other experts of the community. [...] The following functionalities and improvements are included in Release 1.6.0:</p> <ul style="list-style-type: none"> • Remove the verification of the zero-knowledge proofs generated by the control components during the configuration phase. Thereby the verifier no longer has an operational role (in line with measure A.22 of the Federal Chancellery's catalogue of measures). The control components' zero-knowledge proofs are now verified by the setup component in the algorithm CombineEncLongCodeShares. [...] • Incorporate feedback from independent developers of the verifier software with the following: [...] • Simplify the generation of XML files by removing the redundant evoting-decrypt XML and eCH-0110 XML, and the corresponding signature verification algorithms. [...] • Various improvements related to JSON test files and json-schema. [...] • Improve the Verifier Frontend. [...] • Remove the jSurfer dependency (in line with the measure A.10 to reduce third-party dependencies). • Rename algorithm VerifyTallyFiles to VerifyECH0222 and update its operation according to the verifier specification. • In VerifyTallyControlComponentBallotBox, use VerifyDecryption with a loop instead of VerifyDecryptions (GitLab Iss. 21). • Remove the first name from the semantic information of candidates in an election. • Minor code improvements. • Updated dependencies and third-party libraries.
verifier	1.6.1	<p>Release 1.6.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Improved the generation and layout of the election result report. • Improved translation of the verification report. • Minor code improvements. • Updated dependencies and third-party libraries.
verifier	1.6.2	<p>Release 1.6.2 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Improved logging of signature verification by using the OutputToInputStream converter. • Increased number of signatures in the PDF report. • Improved various translations. • Minor code improvements. • Updated dependencies and third-party libraries.

Component	Version	Changes listed in the component's CHANGELOG.md file
verifier	1.6.2.1	<p>Release 1.6.2.1 is a minor maintenance patch containing the following changes:</p> <ul style="list-style-type: none"> • Minor UX improvement to display the dataset information in the verification step. • Updated dependencies and third-party libraries.

References

- [BK22] *Verordnung der Bundeskanzlei über die elektronische Stimmabgabe (VEleS) vom 1. Juli 2022*. Die Schweizerische Bundeskanzlei (BK), 2022.
<https://www.fedlex.admin.ch/eli/cc/2022/336/en>
- [BK23] *Vote électronique, Catalogue of measures by the Confederation and cantons*. Federal Chancellery (FCh), 2023.
<https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/versuchsbuebersicht.html>
- [BK25] *Audit concept v1.6 – For examining Swiss internet voting systems*. Federal Chancellery (FCh), Political Rights Section, 2025.
https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/ueberpruefung_systeme.html
- [Essex25] Alksander Essex. *Rolling Re-Evaluation of the Swiss Post e-Voting System: Version 1.4.5 (Audit Scope 1: Cryptographic Protocol)*, May 30, 2025.
https://www.bk.admin.ch/bk/de/home/politische-rechte/e-voting/ueberpruefung_systeme.html
- [OH25] Philippe Oechslin, Thomas Hofer. *Code, Review of Voting Card Printing Service V3.0*, Version 1.1, Juni 5, 2025.
https://www.bk.admin.ch/bk/de/home/politische-rechte/e-voting/ueberpruefung_systeme.html