Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for informasjonssikkerhet og kommunikasjonsteknologi

Amund Fredrik Strømsnes

Enabling independent audits through verifier specifications

A development framework for verifier specifications in universally verifiable e-voting systems

Masteroppgave i Digital Infrastructure and Cyber security
Veileder: Tjerand Silde
Medveileder: Audhild Høgåsen
Juni 2026

NTNU
Kunnskap for en bedre verden

Amund Fredrik Strømsnes

Enabling independent audits through verifier specifications

A development framework for verifier specifications in universally verifiable e-voting systems

Masteroppgave i Digital Infrastructure and Cyber security
Veileder: Tjerand Silde
Medveileder: Audhild Høgåsen
Juni 2026

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for informasjonssikkerhet og kommunikasjonsteknologi

NTNU

Kunnskap for en bedre verden

Problem description

End-to-end verifiable electronic voting systems provide both individual and universal verifiability. This means that each voter can verify that their own vote has been cast and recorded correctly, and that the election result can be verified as being correct, based on the recorded votes. Universal verifiability is assured through an audit of the election event, during which other desired security properties, such as voter privacy, are also verified.

In the process model for universally verifiable elections by Haenni et al. a verifier is marked as essential component of these audits (E-Vote-ID 2018). Here, the verifier is defined as a software tool used by auditors during an audit of the election event. The verifier is derived through a two-step process of first defining a verifier specification based on the system protocol, and then developing the verifier software based on this specification. However, the process of deriving the verifier specification is not further described, beyond some remarks such as recommending that the verifier specification is made publicly available to enable third-party review and implementation and that the process itself is performed by qualified personnel. This creates an opportunity to explore the development process of verifier specifications further.

The systematisation of knowledge on mechanisms used in verifiable electronic voting systems by Moser, Kirsten and Dörre (E-Vote-ID 2024) presents several different methods of achieving an end-to-end verifiable electronic voting system. However, they note that few systems provide publicly available documentation of how the system protocols and connected functions, such as verifiers, work. This prevents independent reviews of the system protocol to ensure that the claimed security properties, such as verifiability, are provided.

The Swiss Post electronic voting system is one of the systems identified by Moser, Kirsten and Dörre as being well documented, with both a public system specification and a public verifier specification. A new protocol for the Swiss Post system has been announced by Cortier et al. (IACR ePrint 2025/1625) which weakens strong trust assumptions in the setup phase and changes the infrastructure of both the setup and voting phases, but a verifier specification has not yet been developed for the new protocol. This enables us to test the efficacy of a development process model for verifier specifications in a controlled environment, as both versions of the Swiss Post system exist under the same conditions, but with differences in trust distribution and protocol design.

Approved: 2026-02-20 – Associate Professor Silde, Tjerand (NTNU) (Main supervisor)

Abstract

Independent audits are central to universally verifiable e-voting systems. They allow auditors to assess whether the election outcome follows from the recorded votes and whether claimed security properties are supported by the available evidence. However, such audits depend on a well-defined verification process. If the verifier software, audit data, and verification procedures are insufficiently documented, independent parties remain dependent on system vendors and election authorities, even when the system is theoretically verifiable.

We address the development of verifier specifications for universally verifiable e-voting systems. A verifier specification describes what must be verified, which audit artefacts are required, and how verification procedures should be performed. We develop a verifier specification development framework by consolidating recommendations from the literature on election audits, verifier design, documentation, trust assumptions, and third-party implementation. The framework structures the development process from system claims and trust assumptions to verification requirements, audit data, audit methodology, verifier specification structure, and guidance on facilitating independent implementation of verifier software.

The framework is evaluated through a case study of the current and revised Swiss Post e-voting protocols. This case study provides a controlled setting in which the election context remains largely unchanged, while the protocol design and trust distribution differ. The results indicate that the framework can identify audit-relevant consequences of protocol revisions, including changes to verification requirements, audit artefacts, and verification procedures. They also indicate that the framework structure remains applicable across the two evaluated protocol versions, while the contents of the resulting verifier specification change according to the protocol design.

The findings suggest that verifier specifications can help connect theoretically verifiable e-voting systems with practical independent auditing of election events. The proposed framework provides a structured method for developing such specifications, but its results depend on the quality of system documentation, the protocol's evidence-generation model, and the competence of those applying it.

Sammendrag

Tilsyn og valgobservasjon er sentrale i universelt verifiserbare elektroniske valgsystemer. Dette gjør det mulig å vurdere hvorvidt valgresultatet samsvarer med de registrerte stemmene og om påståtte egenskaper, som at stemmer er hemmelige, ble ivaretatt av systemet. Dette forutsetter imidlertid en godt definert tilsynsprosess. Dersom programvaren, datagrunnlaget eller tilsynsprosessen ikke er godt nok dokumentert, blir det vanskelig for uavhengige parter å utføre tilsyn.

Vi undersøker utviklingsprosessen bak det tekniske dokumentet som beskriver tilsynsprosessen og hvordan utviklingen av digitale tilsynsverktøy kan tilgjengeliggjøres for uavhengige parter. Det tekniske dokumentet beskriver hva som må verifiseres, hvilke data som kreves for verifiseringen og hvordan hver verifisering skal utføres. Vi utvikler en definert prosess for å veilede utviklingen av dokumentet, strukturert gjennom et rammeverk. Dette gjøres ved å samle inn anbefalinger fra litteraturen om digital valgobservasjon, tilsynsprosesser og universelt verifiserbare elektroniske valgsystemer. Rammeverket strukturerer utviklingsprosessen fra man definerer systemdesign, påståtte egenskaper og antakelser, til utvikling av spesifikke verifiseringskrav, nødvendig datagrunnlag, tilsynsmetodikk, struktur på dokumentet og hvordan man kan støtte uavhengig implementering av tilsynsverktøy.

Rammeverket evalueres gjennom en casestudie av den nåværende og den reviderte protokollen for Swiss Post sitt valgsystem. Resultatene viser at rammeverkets struktur forblir anvendelig på tvers av de to evaluerte protokollversjonene, mens innholdet i det resulterende dokumentet endres i henhold til protokollens design.

Funnene tyder på at det tekniske dokumentet som beskriver tilsynsprosessen kan bidra til å gjøre systemer som teoretisk sett er verifiserbare til systemer som støtter opp under uavhengig valgobservasjon på en praktisk måte. Samtidig er kvaliteten på det som utvikles ved hjelp av rammeverket avhengig av kvaliteten på tilgjengelig systemdokumentasjon, protokolldesign og kompetansenivået til dem som anvender rammeverket.

Preface

This thesis is the culmination of two years spent in the Master's program in Digital Infrastructure and Cybersecurity at the Department of Cybersecurity and Communication Technology (IIK) at the Norwegian University of Science and Technology (NTNU). The thesis builds especially on the academic knowledge gained through courses on cryptography and the preliminary work performed during the pre-project in the autumn of 2025.

My deepest thanks go to my supervisors, Associate Professor Tjerand Silde at IIK and Audhild Høgåsen at the Swiss Post, who guided me through the final year of research and writing. Their straightforward and valuable feedback throughout the master project has enabled this work to be the best it could be.

I also thank my fellow students for providing good company and a cheery mood in the master's study room, especially those who baked cakes.

Contents

List of Figures

List of Tables

List of Acronyms

DDH Decisional Diffie-Hellman.

DLP Discrete Logarithm Problem.

e-voting electronic voting.

NIZKP Non-interactive zero-knowledge proof.

PKE Public Key Encryption.

SGSP Subgroup generated by small primes.

VEleS Swiss Federal Chancellery Ordinance on Electronic Voting.

ZKP Zero-knowledge proof.

Chapter 1
Introduction

The goal of the project is to research how to achieve a universally verifiable electronic voting (e-voting) system that facilitates independent audits. We pursue our goal through consolidating recommendations from the literature for conducting universally verifiable election audits and facilitating third-party audits. Based on these recommendations, we develop and evaluate a framework to formalise the process for creating necessary technical audit aids. Through the development framework and accompanying recommendations, this research can contribute to the verification process of universally verifiable e-voting systems.

1.1 Motivation

In most democratic countries, elections are conducted using paper ballots, with procedural safeguards, such as private voting booths and public observation of the counting process, giving voters confidence that votes remain secret and are counted correctly. However, these assurances rely on physical transparency and human oversight, which do not naturally translate to digital environments. As a result, e-voting systems must provide equivalent guarantees through technical means, motivating the development of verifiable systems.

A verifiable e-voting system is one that enables voters to confirm that their intended vote has been correctly recorded by the system (individual verifiability), and allows any party to verify that the announced election outcome corresponds to the set of recorded votes (universal verifiability). In addition, the e-voting system must ensure that all recorded votes originate from eligible voters and that no voter is able to vote more than once (eligibility verifiability) [41].

In this work, we focus primarily on universal (and, to some extent, eligibility) verifiability, as these properties depend on the availability and correctness of audit data and verification procedures. In contrast, individual verifiability relies on voter interaction and is therefore only indirectly related to the election audit process.

Universal and eligibility verifiability are realised through the auditing of election

artefacts using verifier software [28, 33, 74]. These audits validate both the correctness of the protocol execution and the resulting election outcome, based on the evidence produced during the election process. This shifts the focus of trust from the voting system alone to the verification process. In particular, the correctness and soundness of the verifier software become critical, as errors in the verification process may lead to incorrect conclusions about the election result.

This concern is closely related to the notion of software independence, defined by Rivest [55] as the property that system errors cannot cause undetectable changes to the election outcome. Subsequent work has argued that this property depends not only on the voting system, but also on the correctness of the verifier software and audit procedures [33]. Therefore, trust in an election outcome depends not only on the security of the voting protocol but also on the integrity and correctness of the entire verification process.

Audits performed by the system vendor lead to greater trust in the vendor [9, 81], whereas independent audits would reduce that trust. One way to facilitate independent audits is for the vendor to provide verifier software [2]. However, an audit is not entirely independent if only vendor-provided verifier software is used [6, 74]. Therefore, to further reduce trust, election authorities and system vendors must also facilitate third-party implementations of verifier software.

The development process for verifiers consists of a two-step process: first, defining a verifier specification based on the system protocol, and then developing the verifier software based on this specification [28]. This process shows that a publicly available verifier specification is a necessary precondition for third parties to implement their own verifier software. Furthermore, independent auditors can verify the verifier's correctness through the specification.

Although the auditability of universally verifiable e-voting systems has been studied before, existing work primarily focuses on audit frameworks and verification mechanisms in isolation. In particular, it provides limited guidance on how systems should be designed, documented, and operated to enable independent audits and third-party implementation of verifier software in practice.

As a result, there is a gap between theoretical verifiability and the ability of independent parties to meaningfully verify election outcomes. This work addresses that gap by constructing a verifier specification development framework, based on the results of a systematic literature review to identify, consolidate, and structure recommendations that support practical auditability and independent verification.

The planned revision of the Swiss Post e-voting protocol [14] provides an opportunity to evaluate the constructed verifier specification development framework. While the e-voting protocol evolves, the election context remains the same, allowing the case study to focus on protocol changes and their impact on the framework's applicability.

1.2 Contributions

Our work consists of three main contributions. First, we identify and consolidate recommendations from the literature on universally verifiable e-voting systems, election audits, verifier specifications, and implementability. These recommendations are organised across six dimensions: verification requirements, audit data, verification processes, trust assumptions, documentation, and implementability. Together, these dimensions describe the main considerations involved in developing verifier specifications that support independent election audits and third-party verifier implementation.

Second, we introduce a framework for developing verifier specifications based on these recommendations. The framework provides a structured process for deriving verification requirements from system claims, security properties, trust assumptions, and protocol design, and for organising these requirements into a verifier specification. In doing so, the framework connects audit design, documentation, and verifier implementability, addressing the gap between theoretical verifiability and the practical requirements for independent auditing.

Third, we evaluate the framework through a case study of the current and revised Swiss Post e-voting protocols. The case study assesses whether the framework remains applicable across two related protocol designs and whether it captures audit-relevant consequences of protocol changes. The results identify changes to verification requirements, audit artefacts, and verification procedures introduced by the revised protocol, and provide guidance on which parts of the verifier specification should be considered during revision.

1.2.1 Research questions

From the problem statement, we have formulated a main objective for the thesis work:

Main objective: To develop a systematic method for deriving and structuring verifier specifications for end-to-end verifiable electronic voting systems by translating security properties, trust assumptions, and protocol-level evidence models into clear, testable, and independently implementable verification requirements.

To achieve this objective, we have further defined four research questions. These are divided into a set of questions that contribute to the development of a systematic method for deriving and structuring verifier specifications (RQ1, RQ2), and a final question that serves as a demonstration and case study of the proposed derivation and structuring model's efficacy (RQ3, RQ4). Answering these questions allows us to both achieve the main objective and demonstrate how to apply the model.

RQ1 How can high-level security properties and trust assumptions be systematically translated into explicit, measurable, and testable verification requirements for verifiable voting protocols?

RQ2 How can these verification requirements be organised into a systematic, modular, and implementation-agnostic verifier specification that supports independent development and evaluation?

RQ3 How do variations in protocol design, trust distribution, and evidence-generation mechanisms influence both the derived verification requirements and the resulting structure of the verifier specification?

RQ4 How effectively does the proposed derivation and structuring method capture and differentiate verification requirements when applied to both the current and revised Swiss Post e-voting system designs?

1.2.2 Scope of thesis

This section covers the scope of the thesis and the assumptions under which the results were achieved.

The field of e-voting includes several areas of study: remote voting, internet voting, on-site electronic voting, and voting using paper ballots and direct-reading electronic machines. The work presented in this thesis addresses all of these areas, as it discusses the election audit process. Therefore, the more general term *electronic voting*, often abbreviated as e-voting, is used in favour of one of the other more restrictive terms.

Most e-voting systems are used in combination with physical ballots in some way, either when the ballot itself is physical [63] or when the election event allows voters to choose whether to vote electronically or with physical ballots [69, 78]. The work described in this thesis does not cover the physical operations related to the election process, but some of the ideas may be applied to systems that employ mixed physical and digital operations, given that the artefacts provided to the auditors, and thus the verifier, are entirely digital.

In the same vein, the property of individual verifiability is partly outside the scope of the thesis, as it requires verification activities by voters themselves; see Section 2.2.3. However, the protocol operations that secure this property on the infrastructure side are included. For example, in the Swiss Post e-voting system, individual verifiability is ensured through return codes. The production of these codes during the setup phase and the process of finding the correct return and finalisation codes during the voting phase may be covered, but not the user's operation of inputting the correct acceptance code.

To prove that the system protocol was executed correctly is a key part of a verifiable e-voting system. This is achieved by generating evidence during protocol

execution, and the specific evidence-generation model is unique to each system. However, the generation of evidence is not directly relevant to the verifier, who is concerned only with the existence and validity of evidence. The evaluation of whether the evidence generated by a system protocol is sufficient to prove any claim by the system developers is not covered. Therefore, we assume that the evidence generated during protocol execution is sufficient to prove all claimed security properties.

Lastly, in this thesis, an election audit refers to the technical verification of digital election artefacts produced during an election event. The audit is performed using verifier software and aims to assess whether the protocol execution, audit data, and election result support the system's claimed security properties. While other audits of artefacts, such as system documentation and source code, are an integral part of the process of verifying e-voting systems, they are covered only where directly relevant. In those cases, the audited artefact is stated explicitly.

1.3 Thesis Organization

The thesis is divided into a preface and several chapters. The preface contains general information about the thesis, such as an abstract, a formal problem statement, and a table of contents. Chapters 1 through 6 define the main contents of the thesis, with each of the chapters' contents described below:

1. **Introduction** – This chapter explains the purpose and motivation behind the thesis, as well as defining the scope.
2. **Background** – The research performed during this thesis builds upon prior research in the field. This background theory is explained within the second chapter.
3. **Methodology** – The methods employed to answer each of the research questions are described in the third chapter. The chapter also explains the reasons behind the different methodological choices.
4. **Results** – Results gathered from performing the stages outlined in Chapter 3 are presented during the fourth chapter.
5. **Discussion** – This chapter discusses the results in Chapter 4 with the goal of answering the research questions posed in Section 1.2.1.
6. **Conclusion** – To conclude the thesis, we summarise the main findings and identify areas for future work.

1.4 Sustainability

This work contributes to goal 10, *reducing inequalities* [76], and 16, *peace, justice and strong institutions* [77], of the United Nations' sustainability goals.

The research contributes to the study and development of verifiable internet-based e-voting systems. These e-voting systems allow independent auditors, who could be

the voters themselves, to audit and verify elections to a level not possible in paper-based elections. This increases transparency in election processes and contributes to target 16.6 [77].

Internet-based remote voting allows any individual with access to an internet-connected device to vote independently, without outside help. This will mean that those with disabilities which hinder their access to voting stations may vote without a proxy, which might compromise their vote privacy. Further research into verifiable internet-based e-voting systems might increase their adoption. The result is that voting becomes more accessible and private for more people, contributing to targets 10.2 [76] and 16.7 [77].

1.5 Related works

Treier [74] and Haenni et al. [28] propose audit frameworks that structure the verification process for universally verifiable elections. However, these works primarily focus on the methodology and organisation of audits, and provide limited guidance on how systems should be designed and documented to support independent audits and third-party implementation of verifier software in practice.

Several works address related aspects of auditability from different perspectives. Moser et al. [48] present recommendations for third parties implementing audit tools for individual verifiability, while Haines et al. [34] define requirements for making e-voting systems auditable from the viewpoint of source code examination. Haenni et al. [27] provide practical recommendations based on their experience developing an e-voting system. These contributions address specific aspects of auditability, including tooling, implementation, and system design.

However, existing work remains fragmented across these perspectives and does not provide a unified set of recommendations that connects protocol design, audit processes, documentation, trust assumptions, and implementability. In particular, limited attention has been paid to how these aspects interact to enable independent, practically feasible verification by third parties.

This work addresses this gap by consolidating recommendations from the literature and creating a framework for developing verifier specifications, enabling the design of e-voting systems that are verifiable not only in theory but also auditable in practice.

Chapter 2
Background

This chapter introduces the key topics needed to gain a fundamental understanding of e-voting systems and the election audit process.

Some sections of this chapter were already presented in the project proposal but have been repurposed for this thesis due to their relevance [64].

2.1 Mathematical preliminaries

Although cryptographic mechanisms are integral to e-voting systems, they are not directly relevant for this thesis. Therefore, this section describes the mathematical preliminaries required to understand the use of cryptographic proofs as a verifiability mechanism. We assume the reader is already somewhat familiar with cryptography.

2.1.1 Hard problems

Most cryptographic schemes can be reduced to one or more hard problems that are infeasible to solve within realistic resource constraints. The Swiss Post e-voting protocol is based on the ElGamal cryptosystem [17] and the relevant hard problems are the Discrete Logarithm Problem (DLP), Decisional Diffie-Hellman (DDH), and Subgroup generated by small primes (SGSP) [70, p. 55].

Discrete logarithm problem. The DLP is to find b, given a, g, and p, such that $a \equiv g^b \pmod{p}$ [25, p. 29]. This is assumed to be infeasible, given p is a *safe prime*, defined as a prime for which $q = \frac{p-1}{2}$ is also prime [25, p. 50]. In elliptic curve cryptography, the discrete logarithm is defined as finding b given a where $a = bP$, where P is a known point [25, p. 67].

Decisional Diffie-Hellman. In the DDH problem, the adversary is given $A = g^a \mod p$, $B = g^b \mod p$ and Z, and must determine whether $Z = g^{ab} \mod p$ or whether $Z = g^z \mod p$, where z is randomly sampled. This is a decisional problem in which only the adversary's advantage in distinguishing between the two cases is important [25, p. 149]. A solver for DLP can be used to solve DDH [25, p. 149].

Subgroup generated by small primes. The third relevant hard problem is the SGSP. SGSP assumes that it is infeasible to distinguish between a random group element and a group element raised to the power of a small prime. Although SGSP is related to DDH, it is a weaker problem that is easier to solve in practice [26].

2.1.2 Zero-knowledge proofs

A Zero-knowledge proof (ZKP) is a cryptographic protocol where a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ of the truthfulness of a statement without revealing any information beyond the validity of the statement itself [25, pp. 407–408]. In simpler terms, the prover can demonstrate knowledge of a secret without disclosing it.

ZKPs provide three properties: *completeness*, meaning an honest prover can always convince the verifier of a true statement; *soundness*, meaning a dishonest prover cannot convince the verifier of a false statement; and *zero-knowledge*, meaning the verifier learns only the validity of the statement [25, pp. 381–408]. Interactive ZKP protocols can be converted into Non-interactive zero-knowledge proof (NIZKP) protocols using the Fiat-Shamir transformation [22]. All proofs later referred to in this work are NIZKP.

There are many areas where ZKPs are applied, but these two types of proofs are the most important for this thesis:

Decryption proofs allow verification that a given plaintext was correctly decrypted from a given ciphertext, without revealing the decryption key to the verifier [25, p. 410].

Shuffle proofs are used to prove that a list of ciphertexts has been correctly permuted and re-encrypted by a mix-net [5]. Such proofs enable verification of the integrity of the mixing process responsible for anonymisation, without revealing the mapping between the input and output to the mixing process [25, pp. 420–422].

2.2 Electronic voting systems

As with paper-based elections, digital elections are governed by an election authority. However, these are performed using an e-voting system provided by a system vendor rather than physical ballot boxes. This section presents the general workings of e-voting systems and the desired properties of such systems, followed by an extended section on verifiability.

Most of this section was already included in the pre-project, but its contents have since been revised to varying degrees [64].

2.2.1 Phases of an e-voting system

The election process using e-voting systems consists of three distinct phases. The setup phase, which aims to create and distribute voting cards; the voting phase, where people vote; and the tally phase, where the votes are counted [25, p. 502].

Setup phase. During the setup phase of an e-voting system, the components used later during the voting process are configured. For example, the cryptographic keys and system components are initialised [25, p. 502]. The participants of the setup phase are the infrastructure participants [25, p. 504], also called system participants [20]. As the setup phase is not time sensitive, multiparty computation is recommended [25, p. 502].

Voting phase. After the setup of the e-voting system is complete, the election authorities can initiate the voting phase. During this phase, eligible voters can cast ballots using a user device running a voting client. The cast ballots are recorded by the system infrastructure in a digital ballot box, with a voting server as the first recipient [25, p. 502].

Tally phase. After the election authorities have closed the digital ballot boxes, preventing further ballots from being cast, the tally phase is initiated. During the tally phase, one or more counters count all recorded ballots cast during the voting phase. Before counting the votes, the ballots are usually anonymised using a mix-net to preserve vote privacy. After the votes have been counted, auditors may verify the tally and issue a certificate of the final result if it is correct [25, pp. 502–503].

2.2.2 Desired security properties of an e-voting protocol

The study of verifiable e-voting has identified several desired properties for e-voting systems. The following list is based on the properties presented by Gjøsteen [25, pp. 489–524]:

Privacy For *privacy* to be provided by an e-voting system, the identity of the voter who gave a specific vote must remain unknown both during and after the election [25, p. 505].

Integrity The *integrity* of an e-voting system is concerned with the published result being consistent with all of the votes cast by the voters and recorded by the voting servers. This is slightly different from the typical definition of integrity in cryptography, which, in this case, would allow for some votes to not be counted as long as they were not altered [25, p. 505].

Verifiability This property is described in Section 2.2.3.

Robustness The *robustness* property is concerned with the system's resistance to an adversary changing the tally result. It is provided when it is hard for an adversary to alter or stop a correct tally from being published [25, p. 507].

Fairness and Eligibility These properties are interconnected and build on the integrity of the tally. An e-voting system is *fair* if no voter can cast more than one vote, meaning all voters have the same number of votes; and the system provides *eligibility* if each recorded vote was cast by a voter with permission to vote [25, p. 507].

Accountability If it is possible to find the responsible participant when a fault is encountered during an election event, *accountability* is provided [25, p. 508].

Coercion Resistance When a voting system is coercion resistant, it is not possible for an adversary to force a voter to vote for a specific option. In physical elections, this can be addressed through voting booths, where only the voter has complete knowledge of what they wrote on the ballot [25, p. 515]. Since the Swiss electoral board allows voting by post, which does not provide coercion resistance, this property will be disregarded for the Swiss Post e-voting system.

2.2.3 Verifiability

While integrity is the e-voting protocol's ability to detect fraudulent changes to the ballot box, verifiability concerns whether processes such as these can be verified during an audit.

An e-voting system is verifiable when it can be confirmed whether the election process was correctly performed and whether the other desired properties were provided during an election event [25, p. 506]. Kremer et al. define a verifiable e-voting system as one that provides [41]:

Individual verifiability enables voters to confirm that their intended vote has been correctly recorded by the system.

Universal verifiability allows any party to verify that the announced election outcome corresponds to the set of recorded votes.

Eligibility verifiability ensures that all recorded votes originate from eligible voters and that no voter has voted more than once.

While individual verifiability relies on voter interaction and is verified directly by voters, universal and eligibility verifiability are realised through the auditing of election artefacts using verifier software [28, 33, 74].

The election artefacts constitute cryptographic proofs demonstrating that the e-voting protocol was executed correctly, as well as other relevant data on the election

context [28]. The result of the election audit is a statement on whether the election process was conducted correctly [25, p. 506], referred to in this thesis as a *verification report*.

The property of verifiability is closely related to software independence, defined as the property that system errors cannot cause undetectable changes to the election outcome [55]. Therefore, when verifiability is ensured through a sound and correct election audit process, voters can be assured that no errors or mischievous actions have altered the tally.

Individual verifiability

Individual verifiability concerns each voter individually. For a system to provide individual verifiability, each voter must be able to verify that the vote recorded by the voting server was the vote they intended to cast, and if they did not cast a vote, a voter must be able to verify that no vote was cast in their name [20]. The two sub-properties of cast-as-intended and recorded-as-cast are covered by what the Swiss Federal Chancellery Ordinance on Electronic Voting (VEleS) defines as individual verifiability [20, Art. 5][1]:

Cast-as-intended is provided when a voter can be assured that the vote received by the voting server was the vote intended to be cast by the voter [25, p. 509].

Recorded-as-cast is the ability for a voter to verify that the vote they intended to cast was correctly recorded. The verification shows the voter that their vote was not rejected or altered after being cast (vote rejection), and that an adversarial voting client or server did not vote on a voter's behalf (vote injection) [70, p. 92]. This should be provided through a cryptographic proof that all listed votes were used during the tally phase [25, p. 509].

There are multiple methods for achieving individual verifiability, but this thesis will only concern itself with *return codes*[2]. When using return codes for individual verifiability, each voter is provided with a set of return codes associated with each voting option before the voting phase, known only to that voter and the voting server. *Cast-as-intended* is provided as follows: after the voter casts their ballot, indicating their intended voting option, they receive a receipt containing that voting option's corresponding return code, which they compare with their complete list of return codes. If the codes are consistent, the voter can be assured that their vote was cast as intended [25, pp. 510–514].

[1] We introduce this definition as it defines the election context of our case study subject, the Swiss Post e-voting system. Furthermore, other sources such as Gjøsteen [25] support subdividing individual verifiability into these two sub-properties.

[2] Individual verifiability is not the main focus of this thesis. Therefore, we have limited the description of individual verifiability mechanisms to what is directly relevant to the case study subject, the Swiss Post e-voting system.

Universal verifiability

Universal verifiability aims to provide auditors with proof that the final tally was correctly computed and not tampered with by an adversary [41]. In a physical voting system, a recount in the presence of independent observers may be performed to confirm that the tally was correctly computed, but in an e-voting system, this must be performed through a combination of cryptographic mechanisms. Achieving universal verifiability requires combining a secret-ballot method with a verifiable-tally method [50].

Secret ballot methods. Moser et al. present two verifiable methods for casting votes in a manner that preserves vote privacy:

Malleable Public Key Encryption (PKE) requires the use of a PKE encryption scheme that provides malleable, also called homomorphic, ciphertexts. The scheme has to support verifiable key generation, distributed decryption, and ZKP of plaintext knowledge and correct decryption [50, pp. 37–46]. The malleability of the ciphertext allows further mathematical operations to be performed on it, meaning it could be processed by the system infrastructure without requiring decryption. As the method supports distributed decryption, the system can be designed to require multiple participants to join together for the complete decryption of a ballot ciphertext [50, p. 40]. Verifiability is ensured by generating keys using a verifiable method and by proofs of knowledge and correct decryption.

Malleable commitments are similar to malleable PKE, enabling further processing of the committed vote, but uses commitments [25, pp. 259–261] instead of PKE. The commitment scheme must support ZKP of plaintext knowledge, and threshold secret sharing, serving the same purpose as distributed decryption [50, pp. 47–52]. As with the malleable PKE method, verifiability is ensured through proofs of knowledge.

Verifiable tally methods. In addition to providing a vote privacy in a verifiable method, the integrity of the tally has to be verified, also without breaking vote privacy. Moser et al. present two such methods:

Homomorphic aggregation performs tallying through adding together the malleable ciphertexts, before decrypting the final aggregate ciphertext. The result of the tally is determined by comparing the aggregate plaintext with a table of all possible tally results. Vote privacy is ensured because it is not possible to identify which ballots contributed to which voting option. Verifiability is ensured in two steps: the aggregation is verified through performing it again, and the tally result is verified through using a proof of correct decryption to verify the aggregate plaintext [50, pp. 53–58].

Verifiable mixing uses a mix-net to anonymise the encrypted ballots by permuting and re-encrypting the ciphertexts, also known as shuffling. After being mixed, the ballots are decrypted. The resulting plaintexts are then processed to determine the tally. Vote privacy is ensured through shuffling, and verifiability is ensured by verifying shuffle proofs [50, pp. 59–64].

2.3 Trust assumptions

In the end, the security of any system has to be based on an assumption of trustworthiness, without any direct proof. The same applies to e-voting systems. We divide these trust assumptions into those related to the underlying cryptography, the software in use, and the system's infrastructure and operation. This division is based on the division found in article 10 of the Federal Chancellery Ordinance on Electronic Voting [20, Art. 10].

Cryptographic assumptions. The security of a cryptographic system is derived by reducing the problem to a known hard problem, which was presented during Section 2.1.1. Although the hard problems are considered infeasible to solve given realistic conditions, it is still a trust assumption.

Software assumptions. The software assumptions concern whether the software in use during the election process can be trusted to be correct and authentic.

Infrastructure and operation assumptions. The trust assumptions regarding the infrastructure and operations concern the use of system participants and communication links at the operational level in an e-voting system. A communication link is assumed to be authenticated, so an adversary cannot modify messages between system participants. However, a general assumption is that an adversary could replay and reschedule messages, given the delivery order remains correct [25, p. 504]. Another general trust assumption is that the link between the voter and the user device can be trusted with respect to privacy [25, p. 504].

2.4 Election audits and the verification process

Election audits are performed to ensure that the system protocol was executed correctly and that all claimed security properties were provided. However, to enable audits of e-voting systems, the protocol execution must produce proofs that can guarantee the system's claims. Verifier software is employed by the auditors to assist them in verifying these proofs [28].

The purpose of the verifier software, also referred to as verifier, is to receive audit data as input, perform a series of verification tests according to a test catalogue, and produce a verification report as output [28]. The audit data includes contextual

information about the election event, evidence generated during protocol execution, and other relevant data for the verification process.

A verifier specification must be available so auditors can correctly use the verifier and assess the completeness of the verification chain produced by the test catalogue. Furthermore, the documentation must provide guidance on developing the verifier to enable software development and verification [28].

Some e-voting systems, such as the Swiss Post e-voting system [20, 72], have a distinct goal of promoting independent implementation of the verifier software. However, none provide proper incentive for independent parties to implement their own verifier software [49].

2.4.1 The development process for verifiers

The development process for verifiers has been briefly described by Haenni et al. [28] as a two-step process: developing a verifier specification based on the system specification, followed by developing the verifier software in accordance with the documentation provided in the verifier specification. This chain of processes shows that the verifier specification and software must be maintained and updated alongside the system protocol [28].

The verifier specification should include the necessary information for developing the verifier software, as well as sufficient information for an auditor to perform election audits [28]. The process of deriving the verifier specification should be performed by personnel competent in cryptography and with great care [28].

To ensure that the verifier software developer's implementation is correct and that the verifier works as intended, test data should be provided alongside the verifier specification [28]. The test data should lead to both successful and failed verifications so that all cases can be verified [74].

Formal verification can be used as an alternative to rigorous testing of the verifier software to ensure its soundness and correctness [33]. This method also serves as an alternative to the more typical software development process, as some formal verification tools allow extraction of a verified executable [31]. However, this development process requires new competencies for the verifier software developer, as they must be familiar with the verification tool and all necessary cryptographic components of the protocol. A lack of competence in these areas may introduce errors, leading to an unsound verification process [52].

2.4.2 The audit and verification report

The purpose of the verifier is to produce a verification report for auditors to use during an audit, based on the verdict of a series of tests in a catalogue [28]. However, verifying only the evidence generated during protocol execution does not guarantee

that the evidence itself is complete, internally consistent, or even generated during that election event. Therefore, the provided audit data itself must be verified [28]. We can provide the following categorisation of verification requirements [28]:

Completeness The election process data presented to the verifier can be confirmed as complete if the available data covers the entire election process and enables complete verification of the event [28]. The goal is to establish that no data was omitted from the dataset.

Authenticity The authenticity of the election process data concerns whether it can be unambiguously confirmed that the data's creator was the authorised participant [28].

Consistency To ensure the data is in the correct format, the verifier checks the dataset's consistency by verifying that related elements use the same names and formats, and that the information is consistent [28].

Integrity The integrity category verifications confirm that the elements in the election process data correspond to those in the specification [28].

Evidence The last category the verifier checks is evidence. These verifications concern whether the proofs provided in the election process data are valid and correct, and whether the evidence provided can assure us that the protocol steps were performed correctly [28].

Chapter 3
Methodology

The main objective of the work, stated in Section 1.2.1, is addressed by creating a framework for developing verifier specifications, consisting of recommendations drawn from the relevant literature on auditing elections using verifiable systems. To compile the list of recommendations, we conduct a literature review to extract recommendations from existing guidelines and practices.

The culmination of the methodology is a framework for developing verifier specifications, presented in Section 4.2, evaluated through a case study of the Swiss Post e-voting system. The goal of the verifier specification development framework is to produce a verifier specification, given the system's protocol design, claimed security properties, and the trust assumptions under which they are to be provided. The framework is created by identifying the main steps of the development process and organising the recommendations by the step they support. Each identified framework step is explained in an overview, with a figure illustrating the process and its inputs.

Each of the four research questions, presented in 1.2.1, serves a different purpose in the creation and evaluation of the verifier specification development framework. RQ1 and RQ2 define the framework's intent and serve as evaluation metrics to assess whether the recommendations from the literature review cover the necessary steps for effectively developing verifier specifications. RQ3 and RQ4 evaluate the framework through a case study of the Swiss Post e-voting system, examining the extent to which the framework steps are affected by differences between trust models and protocol designs.

3.1 Literature review

The literature review conducted in this work follows systematic principles to ensure transparency, reproducibility, and comprehensive coverage of relevant research. The approach consists of a structured search over a predefined set of databases using carefully constructed queries to obtain an initial corpus of records. This corpus is subsequently subjected to a multi-stage screening process, after which the remaining works are selected for detailed evaluation.

Table 3.1: Literature review search query components

Category	Keywords
E-voting systems	electronic voting, e-voting, evoting, internet voting, i-voting, ivoting, online voting, cryptographic voting, remote voting, digital voting, verifiable voting, verifiable election*, voting system
Verifiability and audits	verifi*, audit*
Implementation and practice	best practice, lessons learned, implementation security, implementation recommendation, mistaken implementation*, reference implementation*, compliance, regulatory framework, assurance*, source code examination, technical standard, verification capabilit*, performance requirement, formal verification, vulnerabilit*, verification process, verification software

We queried three databases chosen to provide broad coverage of both e-voting and cryptographic research: the IACR Cryptology ePrint Archive[1], which hosts cryptographic preprints, and Web of Science[2] and Scopus[3], which index peer-reviewed publications across a wide range of disciplines.

Each database was queried using the same search string. Searches were restricted to the title, abstract, and keyword fields, and were constructed from keywords grouped into three thematic categories, as shown in Table 3.1. Within each category, keywords were combined using the Boolean operator OR, while the categories themselves were combined using AND.

To ensure precise matching, keywords were enclosed in quotation marks, except where wildcard characters were used. The wildcard symbol (*) was employed to capture multiple word forms sharing a common root; for example, *verifi** matches *verifier*, *verifiable*, and *verification*. The three keyword categories are defined as follows:

E-voting systems This category targets literature on electronic voting systems. As the field employs varied terminology, multiple synonyms, spellings, and related terms were included to maximise coverage.

Verifiability and audits This category ensures that retrieved records address verifiability or auditing. Some overlap may occur with terms such as *verifiable election* or *verifiable voting* from the first category; however, such overlap is retained to avoid excluding relevant works.

[1] https://eprint.iacr.org
[2] https://www.webofscience.com
[3] https://www.scopus.com

Implementation and practice This category encompasses literature on implementation aspects, including verification processes, standardisation, and known implementation issues. It also includes terms such as *best practice* and *lessons learned*, which aim to provide experience-based, practical guidance.

After obtaining the initial corpus of records, we conducted a multi-stage screening process to identify relevant works. This process first removed duplicate entries, then filtered based on title and abstract, and finally conducted a full-text review of the remaining records. In cases where multiple versions of the same work were retrieved, only the most recent version was included.

In total, 355 records were identified, of which 105 were duplicates. Of the remaining records, 98 were deemed irrelevant based on title and abstract screening, and a further 100 were excluded following full-text review.

The relatively high number of excluded records indicates that the search query may be broad. However, a broader query was intentionally chosen to maximise coverage and reduce the risk of omitting relevant literature, favouring recall over precision in the initial search phase.

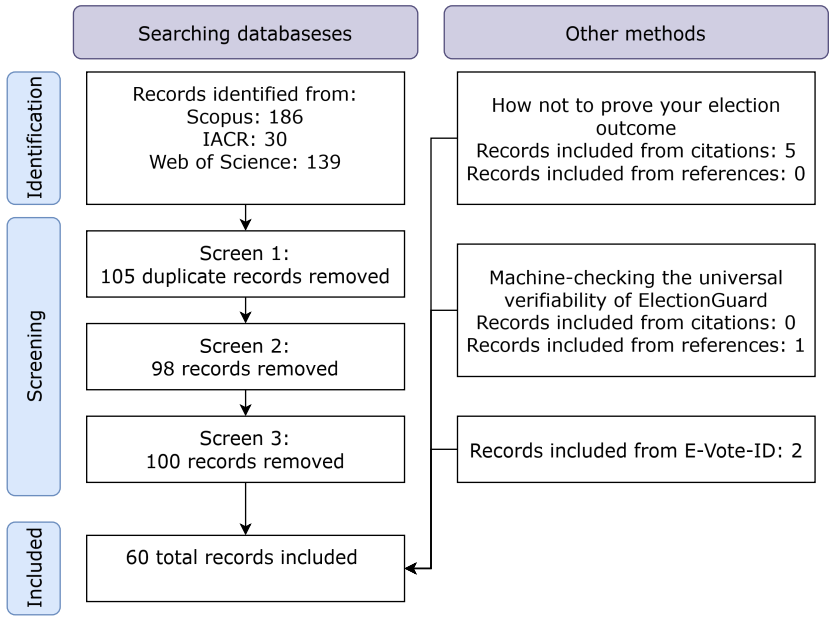

Figure 3.1: Overview of the results from the literature review.

To complement the resulting set of records and ensure broad coverage, two additional sources were incorporated after the screening process. First, all published

long- and short-form papers from the E-Vote-ID conference[4] were evaluated for relevance and included in the final dataset when applicable. These papers were considered separately due to the venue's high relevance to e-voting research. This step resulted in the inclusion of two additional records.

Second, we performed backwards and forward reference searches based on two works identified as particularly influential within the corpus, namely *How not to prove your election outcome* [30] and *Machine-checking the universal verifiability of ElectionGuard* [31]. This process yielded six additional relevant records.

An overview of the complete literature selection process is presented in Figure 3.1. The findings are organised into categories that reflect key aspects of universally verifiable e-voting systems and the associated audit process.

The categories were chosen because they represent different aspects of the audit process: four relate to the general audit process for universally verifiable elections, and the last two capture recommendations for how to facilitate third-party audits. The information we identified regarding what needs to be verified is organised under the category *verification requirements*. The category *available audit data* concerns the audit data presented to the auditors. For this category, we aim to determine which type of information auditors should receive. The category *verification process* aims to capture information regarding the audit and verification process itself. This includes the timing of the audit, specific guidance on how to conduct it, or remarks about the auditors. There always exist underlying trust assumptions and an associated threat model for e-voting systems. Therefore, the category *trust assumptions* seeks to identify which assumptions about trustworthiness, and their associated pitfalls, are made regarding the audit process.

A central part of the development of verifier software is the verifier specification, as noted in Section 2.4. Therefore, we have constructed the category *documentation* to compile recommendations for developing a verifier specification that facilitates independent audits. The category *implementability* builds further towards the goal by identifying specific remarks on facilitating third-party audits and the independent implementation of a verifier. Table 3.2 summarises these categories, together with the number of records contributing to each.

Individual records may be assigned to multiple categories where they address more than one aspect of the audit process. Within certain categories, findings are further grouped to improve clarity and presentation.

Each category concludes with a discussion that synthesises the results, resolves differences in perspective across the literature, and highlights tensions where competing goals are supported by conflicting methods or assumptions.

[4] https://e-vote-id.org

Category	Focus	No. of records
Verification requirements	What must be verified	40
Available audit data	What information auditors receive	33
Verification process	Audit methodology and procedure	25
Trust assumptions	Trust model of audits	9
Documentation	Structure of documentation	19
Implementability	Facilitating third-party audits	22

Table 3.2: Overview of categories and included records per category.

3.2 Verifier specification development framework

We propose a framework for developing verifier specifications based on the recommendations identified during the literature review. Using these recommendations, we identify the main steps of the verifier specification development framework and organise the recommendations by the framework step they support. The framework defines a process that takes as input the protocol design, the security properties claimed by the system, and the trust assumptions under which these properties are provided, and outputs a verifier specification. These inputs are identified from the system specification and related documentation when applying the model.

3.2.1 Deriving verification requirements

RQ1: *How can high-level security properties and trust assumptions be systematically translated into explicit, measurable, and testable verification requirements for verifiable voting protocols?*

The purpose of RQ1 is to define the derivation component of the proposed framework. Specifically, it seeks to establish a systematic process for transforming the security properties and trust assumptions of an e-voting system into concrete verification requirements that can subsequently be implemented and audited. The method outlined in Section 3.1 provides the foundation for answering this research question. In particular, the category *verification requirement* identifies which verification requirements are needed.

The term *verification requirement* is integral to this research question. We define a verification requirement as a measurable and testable condition that must be verified during an election audit to establish confidence that a claimed system property was provided during that election, for example, the correctness of the tally or voter privacy.

We evaluate the framework's process for deriving verification requirements using RQ1 as the evaluation metric; this is discussed in Section 5.2.1. The aim is to determine whether the framework provides a systematic method for deriving explicit,

measurable, and testable verification requirements. We also examine whether the verification requirements provide a complete and sound verification chain, and whether some can be met using readily available tools and methods.

3.2.2 Structuring the verifier specification

RQ2: *How can these verification requirements be organised into a systematic, modular, and implementation-agnostic verifier specification that supports independent development and evaluation?*

While RQ1 concerns the derivation of verification requirements, RQ2 concerns their organisation into a verifier specification for use by auditors, verifier developers, and other stakeholders involved in the audit process.

This component of the framework transforms the verification requirements into a coherent verifier specification that facilitates independent audits and the implementation of third-party verifiers through its structure and the information it contains. The steps required to achieve this objective are to identify the required audit data and the information to be documented, and define the audit methodology. What each of these steps entails is based on the supporting recommendations identified in the literature review results. However, for the verifier specification and the system vendor to support independent development and evaluation, a final framework step is required to present recommendations to facilitate audits.

To answer this question, the recommendations identified in the literature review are analysed with particular emphasis on the categories *available audit data*, *verification process*, and *documentation*. Although all categories of recommendations, presented in Table 3.2, provide relevant input for the content and structure of the verifier specification, these three categories are identified as the most relevant due to defining the key steps of identifying the necessary audit data, defining the audit methodology and recommendations for the documentation itself. The final framework step, concerning the facilitation of audits and the implementation of third-party verifiers, is constructed by analysing all recommendations and identifying those that influence the audit and implementation processes of the verifier specification itself.

These framework steps are evaluated in Section 5.2.2. The evaluation is based on RQ2 and assesses whether the proposed method produces verifier specifications that are modular, implementation-agnostic, and support the independent implementation of verifiers. Furthermore, we discuss whether the resulting verifier specification preserves traceability between system claims, verification requirements, audit artefacts, and verification procedures.

3.3 Case study

The final step in this work's methodology is to conduct a case study using the verifier specification development framework. The purpose of the case study is not primarily to derive a verifier specification, but to evaluate how the framework captures differences between two e-voting protocols. The Swiss Post system is used as a realistic, well-documented case study through which the framework can be applied to a practical e-voting system and to a subsequent protocol revision.

There are several reasons why the Swiss Post e-voting system is a suitable candidate for evaluating the framework. As remarked by Haines et al., a top-of-the-line system must be used to gain the most from analyses [35]. The Swiss Post e-voting system has been identified as one of the few well-documented verifiable e-voting systems with publicly available documentation, and is the sole system under regular independent review among such systems [49]. Second, a revised protocol with an altered trust distribution has been presented [14], enabling us to analyse the framework in the same election context, but for different protocol designs. Therefore, the Swiss Post e-voting system is a strong candidate for evaluating the framework.

The case study consists of three distinct stages, with the first stage describing both the current and revised Swiss Post e-voting system. The aim is to describe the election context, the claimed security properties, the trust assumptions, and the protocol designs for the current and revised system versions. The current election audit methodology is also presented as a reference point for the later stages. The description of the two systems is presented in Section 4.3.1.

The results of the first stage are based on the proposed protocol by Cortier et al. [14] and the publicly available documentation for the Swiss Post e-voting system, including system and verifier specifications, as well as other relevant information on their GitLab webpage[5]. As the system operates within a regulatory context, the analysis also incorporates the requirements imposed by the Swiss Federal Chancellery through the VEleS [20].

The second stage is to apply the verifier specification development framework, and is presented in Section 4.3.2. Each framework step is carried out by addressing each supporting recommendation and stating how the changes between the current and revised systems respond to it. The impact of these changes is compared with the current audit methodology and verifier specification, as presented during the system description. The comparison aims to determine how protocol changes affect the resulting verification requirements, what modifications are required to the verifier specification, and whether any new verification tasks emerge.

The final stage of the case study is performed as a discussion during Sections 5.3.1 and 5.3.2. There, we discuss RQ3 and RQ4, drawing on the development framework itself and the case study results, to assess the framework's generality and evaluate

[5] https://gitlab.com/swisspost-evoting/

the extent to which its steps are affected by differences in trust models and protocol designs.

To conclude, three distinct results are obtained from the case study. First, it provides a practical evaluation of the framework's ability to derive and structure verification requirements from a real-world e-voting system. Second, it demonstrates how changes in protocol design translate into concrete changes in audit procedures and verifier specifications, illustrating the framework's usefulness during system updates. Third, it identifies what demands the proposed changes to the Swiss Post e-voting system impose on the election audit process.

3.3.1 RQ3: General applicability of framework

RQ3: *How do variations in protocol design, trust distribution, and evidence-generation mechanisms influence both the derived verification requirements and the resulting structure of the verifier specification?*

The goal of RQ3 is to ensure that the list of recommendations in Section 4.1 constitutes a widely applicable framework rather than being tailored to a specific protocol design. This is achieved by discussing the results of stage two of the case study and evaluating the recommendations from the literature review, using RQ3 as the evaluation metric. This evaluation is performed as a discussion in Section 5.3.1 and covers three main topics:

The first topic concerns whether the framework favours a particular set of mechanisms for universal verifiability. To evaluate whether the framework is independent of verifiability mechanisms, we discuss how the framework steps are influenced by the revision of the Swiss Post e-voting system, and what impact the verifiability mechanisms presented in Section 2.2.3 could have.

The second topic concerns whether certain trust assumptions influence which recommendations to prioritise, or whether any are deemed irrelevant. We discuss the recommendations of category *trust assumptions*, presented in Section 4.1.4, and the impact of the revised trust model of the Swiss Post e-voting system.

The last topic concerns how the protocol's evidence-generation model affects the verification process and the framework. We discuss the impact of the revised protocol design of the Swiss Post e-voting system, with a focus on the demands imposed on evidence verification.

By discussing these three topics, we aim to assess the extent to which the framework is independent of protocol design, trust distribution, and evidence-generation mechanisms.

3.3.2 RQ4: Applicability during system revision

RQ4: *How effectively does the proposed derivation and structuring method capture and differentiate verification requirements when applied to both the current and revised Swiss Post e-voting system designs?*

The objective of RQ4 is to determine whether the verifier specification development framework can capture and differentiate verification tasks across systems with different trust models and protocol designs. This ability determines the extent to which the framework accurately reflects protocol changes, identifying its usability during protocol updates.

As with the prior research questions, the answer to RQ4 is determined through discussion in Section 5.3.2. There, we identify the extent to which the framework introduces new, or modifies existing, verification activities and requirements in response to the system revision. The current audit methodology of the Swiss Post e-voting system, presented in Section 4.3.1, is used to discuss the extent of the modifications and whether some verification tests become obsolete.

By discussing the necessary changes to the Swiss Post e-voting audit methodology identified through the case study, we evaluate the framework's ability to provide accurate guidance during system revision. Together with RQ3, this determines the generality and applicability of the introduced verifier specification development framework.

3.4 Review of methodology

The methodology is designed to produce results sufficient to answer the research questions listed in Section 1.2.1. However, some trade-offs must be acknowledged, as well as potential alterations for future research.

During the literature review, we used a broad query, which retrieved 250 unique works. However, only 52 of these were considered relevant. As described in Section 3.1, a broad query was preferred as it increases the likelihood of capturing all relevant works, at the cost of a higher workload. An example of the literature review's coverage is the number of works identified through alternative methods. Only eight papers, not already in the retrieved list of works, were considered relevant. However, it is entirely possible that a more precise query would yield the same relevant works while returning fewer irrelevant ones. Therefore, further refinement of the query string should be considered if the literature review were conducted anew.

The research questions are addressed through discussion, using the questions themselves as a qualitative evaluation metric for evaluating the literature review results. However, a quantitative method with highly detailed evaluation metrics would also be possible, enabling granular results that make comparisons between frameworks easier and reduce the risk of human error during evaluation. Still, this

granularity comes at the cost of increased effort in creating the evaluation metrics. Furthermore, the literature review serves as a single point of failure for the project, and using a quantitative method to evaluate the results was not considered sufficient to reduce the risk of human error to a level worth the increased workload. In addition, all other steps of the methodology follow a qualitative approach, creating internal consistency. Therefore, to maintain a consistent methodological structure, the research questions were chosen as the metric for evaluating the framework derived from the literature review.

Chapter 4
Results

This chapter is divided into three main sections. Section 4.1 presents the results of the literature review defined in Section 3.1. The second section presents the verifier specification framework, developed through the methodology outlined in Sections 3.2.1 and 3.2.2. The final section presents the Swiss Post case study, beginning with an overview of the system in Section 4.3.1 and then applying the development framework in Section 4.3.2.

4.1 Literature review

Based on the literature, we identify practices and guidelines for conducting election audits and supporting independent verification, following the methodology outlined in Section 3.1. This section presents the findings organised into six categories, each capturing a distinct aspect of the audit process. Together, these categories provide a structured view of the requirements and considerations necessary for achieving practical auditability. An overview of the number of recommendations per category is found in Table 4.1.

Category	**No. of Recommendations**
Verification requirements	2
Available audit data	2
Verification process	6
Trust assumptions	3
Documentation	6
Implementability	6

Table 4.1: Overview of the number of recommendations per category.

4.1.1 Verification requirements

We organise the identified requirements into four groups, reflecting different aspects of the e-voting system and audit process.

System and audit correctness. This group concerns requirements related to the correctness and soundness of the e-voting system, the audit process, and the verifier software:

1. The system specification and implementation should be audited to demonstrate that the system provides its claimed security properties [1, 16, 46, 61, 68].
2. The audit process itself should be assessed to establish its correctness and soundness [1, 19].
3. The verifier software should be validated to ensure its correctness and soundness, as errors may lead to incorrect conclusions about the election result [19, 31–33, 65, 74].

Protocol execution and evidence. This group concerns requirements related to the evidence generated during protocol execution:

4. The audit should confirm that the protocol was executed correctly [11, 19, 28, 58, 60, 65, 74].
5. Post-election audits should confirm that no unresolved voter complaints were raised through individual verifiability mechanisms [16, 44].
6. The audit of the tally should ensure that no invalid votes are included in the final result [2, 11, 16, 18, 37, 46, 53, 54, 59, 61, 81].
7. The correctness of the protocol execution, tally, and claimed security properties should be established by evaluating the evidence and proof transcripts[1] generated during execution [11, 19, 28, 29, 31, 32, 37, 44, 45, 59, 80, 81].

Audit data. The third group concerns the integrity and quality of the audit data:

8. The integrity of the audit data should be checked to ensure that it has not been modified or tampered with [2, 8, 10, 12, 28, 53, 60, 67, 75].
9. The authenticity of the audit data should be established to ensure that it originates from the expected system components [28, 74, 75].
10. The completeness and consistency of the audit data should be ensured, confirming that all required elements are present and mutually consistent, including public keys and other system parameters [28, 35–37, 46, 58, 74, 75].

Limitations and pitfalls. The final group highlights issues that may prevent verification procedures from achieving their goals:

[1] Proof transcripts may include, for example, proofs of correct decryption, mixing, or other cryptographic operations, depending on the protocol design.

11. The evidence model of the e-voting protocol may be insufficient to support its claimed security properties [7, 30, 36, 68, 75].
12. Errors in the system implementation may invalidate security guarantees regardless of the evidence produced [24, 30, 44, 47].
13. The verification process may be incomplete, either by considering only part of the available evidence or by failing to cover all relevant components of the system [15, 27, 30].

Discussion

If evidence-generation mechanisms and audit procedures are properly designed and aligned, evaluating the resulting evidence can provide strong assurance of correct protocol execution and tally. However, this assurance depends critically on the quality of the audit data. In particular, its integrity, authenticity, completeness, and consistency must be established to ensure that the evidence is trustworthy and relevant to the election.

These guarantees hold only within the assumptions of the underlying threat model. Implementation errors, incomplete evidence models, or flaws in the verification process may undermine the validity of the audit, emphasising the need for careful system and audit design.

Recommendations

We derive the following requirements for election audits:

1A Establish the correctness and soundness of the e-voting protocol, the audit process, and the verifier software.

1B Ensure the validity of the audit data to support correct interpretation of the protocol execution:
   a) Assess the evidence produced during protocol execution.
   b) Ensure completeness of the audit data.
   c) Ensure integrity of the audit data.
   d) Establish authenticity of the audit data.
   e) Ensure consistency of the audit data.

4.1.2 Available audit data

For auditors to conduct meaningful and effective audits, they must be provided with sufficient information about both the election outcome and the underlying system. The literature identifies several categories of information that should be made available:

1. Access to the source code of the e-voting system and the associated verifier software is widely considered essential for enabling thorough and independent audits [27, 34, 39, 46, 54, 81].
2. Auditors should be provided with comprehensive system documentation to enable understanding of system behaviour, design decisions, and security properties [16, 36, 68].
3. The final tally and the corresponding ballot box must be accessible to auditors to enable verification of tally correctness [3, 4, 10, 11, 28, 35, 46, 51, 54, 61, 75, 80, 81]. This should be complemented by comprehensive audit logs from monitoring systems, individual components, and intermediate processing steps, allowing the audit to reconstruct and analyse the execution of the election process [19, 43, 46, 48, 58, 59, 79].
4. Audit data should include proof transcripts and other cryptographic artefacts generated during protocol execution, enabling verification of the correctness of the protocol and its underlying cryptographic operations [12, 15, 31, 46, 53, 54, 57–60, 75].
5. Auditors should be provided with the inputs and outputs of the election process, together with sufficient metadata[2] to allow correct interpretation and validation of the audit data [1, 12, 28, 40, 46, 57, 75].

Discussion

The literature shows a high degree of agreement regarding the types of data that should be made available to auditors. Beyond protocol evidence, effective auditing requires comprehensive supporting information, including system documentation, audit procedures, and descriptions of audit artefacts, to enable correct interpretation and validation of the data.

These findings highlight that the availability of audit data alone is insufficient: the data must also be complete, well-structured, and accompanied by adequate contextual information. Without this, auditors may be unable to correctly interpret the evidence or verify the claimed properties of the system, limiting the practical effectiveness of the audit process.

Recommendations

We derive the following recommendations regarding the information that should be made available to auditors:

2A Provide complete audit data, including the final tally, intermediate results, and proof transcripts, together with sufficient metadata (e.g., checksums, digital

[2]This includes, for example, randomisation seeds, authentication data such as cryptographic keys, and contextual information about the election, including ballots and candidates.

signatures, and election context information) to ensure the data can be correctly interpreted and validated.

2B Provide comprehensive system documentation and access to the source code of both the e-voting system and the verifier software to enable thorough analysis and independent verification.

4.1.3 Verification process

The findings related to the verification process cover both the methodology and tooling used to conduct audits. These requirements can be grouped into three main aspects: audit timing, audit execution, and audit reliability.

Audit timing and phases. The literature consistently recommends that audits should be performed at multiple stages of the election process:

1. An initial audit should be conducted after the setup phase, before voting begins, to verify that the system has been correctly initialised before any votes are cast [2, 6, 11, 13, 15, 19, 29, 39, 46, 57–59].
2. After the tally phase, the correctness of the tally and the execution of the protocol should be verified [4, 16, 19, 24, 29, 39, 53, 54, 60, 61, 67, 68].

When verification is performed after the tally phase, strict procedural ordering of tests is not required, enabling optimisations such as batch verification [39] and parallel execution [1]. Also, probabilistic techniques, such as random sampling, may be used to verify tally correctness in certain settings [2, 19, 23, 59].

Audit execution and reporting. The following requirements concern how audits are conducted and reported:

3. Audits should be performed using dedicated verifier software to ensure systematic and reproducible verification [19, 23, 24, 31, 32, 68, 74, 75].
4. The audit process should produce a verification report documenting the results [28, 75].
5. To support accountability, the report should, where feasible, describe identified failures and quantify their extent [28].
6. Audits should be conducted by independent entities and involve relevant stakeholders to enhance transparency and reduce reliance on individual parties [16, 38, 74].
7. Auditors should possess sufficient expertise in cryptography and the underlying system to correctly interpret the verification process [37, 74].

Limitations and reliability of the verification process. The literature also highlights several factors that may compromise the soundness of the verification process:

8. The verification process may fail if not all planned audit steps are carried out [75].

9. Errors in the development of verifier software may introduce faults that remain undetected during audits [31].
10. Even when formally verified, the verification process itself may rely on assumptions or tools whose correctness is not guaranteed [52].

Discussion

The literature presents a largely consistent view of the verification process, emphasising structured audit phases, the use of dedicated verifier software, and the importance of comprehensive reporting. However, a key challenge lies in the soundness of the verification process itself.

In particular, the correctness of an audit depends not only on the availability of evidence, but also on the reliability of the tools and procedures used to verify it. This introduces a recursive dependency: the verifier software and audit methodology must themselves be trusted or independently validated. As a result, the verification process forms part of a broader chain of trust, in which weaknesses at any stage may undermine the overall assurance provided by the audit.

These findings highlight that achieving meaningful verification requires careful design of both the audit process and the supporting tooling, as well as explicit consideration of the assumptions underlying their correctness.

Recommendations

We derive the following recommendations for structuring and executing the verification process:

3A Perform an audit after the setup phase, prior to the start of voting, to ensure that the system has been correctly initialised. Voting should only proceed if no errors are detected.

3B Perform an audit after the tally phase to verify the correctness of the protocol execution and the final election outcome.

3C Conduct audits using verifier software that is demonstrably correct and sound, as flaws in the verifier may lead to incorrect conclusions.

3D Ensure that the verification process produces a comprehensive audit report. In the event of verification failure, the report should clearly describe the nature and extent of the failure.

3E Ensure that both developers of verifier software and auditors using it possess sufficient expertise in cryptography and the underlying e-voting system.

3F Involve independent entities and relevant stakeholders in the audit process to enhance transparency and reduce reliance on any single party.

4.1.4 Trust assumptions

All systems operate within a trust model, composed of trust assumptions that affect the e-voting system, verifier software, auditors, and audit infrastructure. The findings below concern the audit trust model and how trust assumptions interact with other security properties:

1. Trust assumptions on the auditors, the verifier software and the method that validates it, should be explicitly defined rather than implicitly assumed [58, 74].
2. Audit procedures may rely on either deterministic or probabilistic trust models, depending on the verification task being performed [56].
3. Trust assumptions could limit the verifiability of the system, but may be introduced to preserve other security properties, such as voter privacy, when direct verification would compromise them [56, 74].
4. Laws, regulations, and election-specific operational constraints may affect how the audit trust model is constructed [16, 29].
5. Independent audits should be performed to reduce the trust placed in the system vendor [9, 81].
6. A verification test does not provide any guarantees when it does not align with the trust model [9, 47, 75].

Discussion

One identified tension concerns audit independence and reliance on trusted parties. Trusting auditors, verifier software, or vendor-controlled infrastructure can simplify the audit process and reduce complexity. However, concentrating trust in a limited set of actors weakens the audit's practical independence and increases the consequences of compromise or dishonesty.

A similar identified tension concerns transparency and audit independence. Vendor-provided verifier software and audit tooling facilitate audits, but reliance on vendor-controlled tools weakens the independence of the verification process. The vendor should therefore support independently developed verifier implementations to reduce this concentration of trust.

Finally, the literature identifies tensions between deterministic and probabilistic trust models. Some audit procedures provide deterministic guarantees, while others rely on probabilistic assurance derived from sampling methods or cryptographic hardness assumptions. Verification methods must therefore remain consistent with the underlying trust model.

Recommendations

We derive the following recommendations regarding trust assumptions:

4A Clearly define the trust assumptions for the e-voting system, audit process, auditors, verifier software, and supporting infrastructure.
4B Reduce trust in the system vendor through facilitating independent audits and the development of third-party verifier software.
4C Ensure that verification procedures remain consistent with the selected audit trust model.

4.1.5 Documentation

System documentation provides auditors with an understanding of the e-voting system, while audit documentation defines how verification should be conducted. Together, they enable auditors to correctly interpret audit data and ensure that the verification process is sound. The literature identifies several requirements for documentation, which we group into four categories.

Documentation of security properties and models. The first group concerns how system security properties and assumptions are specified:

1. Documentation should clearly state the security properties claimed by the system [34, 48, 68], and define them using precise and appropriate formalism or terminology [4, 8, 37].
2. Documentation should explain how these security properties are achieved, including the mechanisms and assumptions underlying them [4, 35, 37].

Documentation of system behaviour. The second group concerns the description of system behaviour and structure:

3. Documentation should include both high-level and low-level descriptions of the protocol, system components, and overall election process [27, 28, 48, 65, 68].
4. All system algorithms should be specified in detail, preferably using pseudocode [16, 19, 27, 34, 37, 39, 48].

Algorithm specifications should define inputs, outputs, and execution context [48], decompose complex procedures into smaller components where appropriate [48], and clearly specify data formats and serialisation methods [16, 19, 27]. Relevant cryptographic parameters should also be explicitly defined [27].

Documentation of the verification process. The third group concerns how the audit process itself is specified:

5. Documentation should define the audit process and its outputs, with a focus on the verification tasks performed by the verifier [28].
6. The audit should be formalised as a structured set of verification tests, organised in a test catalogue, where each test corresponds to a specific verification requirement [28, 74].

7. The roles, responsibilities, and interactions of all parties involved in the audit process should be clearly specified [11, 19, 29, 36, 37, 48, 75].
8. Where applicable, documentation should specify the possible degrees of failure for each verification test [28].

Documentation maintenance and consistency. The final group concerns ensuring that documentation remains reliable and aligned with the system:

9. The documentation, source code, and executable implementations should remain consistent with one another [27, 28, 31, 34, 36, 48, 68].
10. Documentation and implementation should be continuously maintained and updated as flaws are identified and corrected [48].

Discussion

The findings indicate that comprehensive documentation of the e-voting system, verifier software, and audit process is essential for enabling effective auditing. However, a fundamental trade-off arises between the level of detail and the maintainability of the documentation. While detailed and precise documentation improves clarity and supports correct verification, it also increases the effort required to ensure consistency with the evolving system.

Maintaining alignment between documentation, source code, and system behaviour becomes increasingly challenging as systems evolve, potentially introducing inconsistencies that undermine the reliability of the audit process. Although the literature proposes methods to improve readability and structure, such as modular descriptions and the use of pseudocode, these approaches primarily address presentation rather than reducing the underlying maintenance burden.

These findings highlight that documentation quality is not solely a matter of completeness, but also of sustainability. Ensuring that documentation remains accurate, consistent, and up to date over time is a critical requirement for supporting reliable and independent verification.

Recommendations

We derive the following recommendations for system and audit process documentation:

5A Clearly document the system's claimed security properties, threat model, trust assumptions, and the mechanisms by which these guarantees are achieved.

5B Document the complete protocol structure, including algorithms, cryptographic mechanisms, data structures, component interactions, and the overall election process.

5C Specify the audit process in detail, including verifier responsibilities, verification procedures, and how individual tests collectively establish a complete verification chain.

5D Ensure that documentation, source code, and deployed systems remain consistent and are continuously maintained and updated, so that documentation remains accurate and reliable over time.

5E Improve clarity and reduce ambiguity by decomposing complex algorithms, using readable pseudocode, and defining explicit data formats and interfaces.

5F Organise verification tests in categories according to the corresponding verification requirement.

4.1.6 Implementability

Achieving independent verification in practice requires more than theoretical auditability: election authorities and system vendors must actively support third-party auditors and verifier implementers. The findings highlight several requirements for enabling practical and meaningful independent audits.

Support for independent auditors. The first group of findings concerns how election authorities and system vendors can facilitate independent audits:

1. Independent auditors should be permitted to audit the election and provided with access to all non-confidential artefacts and relevant outputs [46, 74].
2. Auditors should be given sufficient time to perform their work, while allowing adequate time for system vendors to address any identified issues [16, 36, 74].
3. Verifier software should be made available to support audit execution [2]. However, reliance solely on vendor-provided tools reduces audit independence, as it introduces a dependency on the same entity being evaluated [6, 74].
4. Documentation of system algorithms and design should be publicly available to enable independent understanding and verification [7, 15, 16, 62].
5. Data and documentation should be presented in a form that is accessible and understandable to potential auditors [68].

Verifier development and usability. The second group concerns requirements for enabling third-party development and validation of verifier software:

6. The barrier to entry for third-party implementers should be minimised by reducing the complexity of the verifier and verification process, for example by favouring simpler designs and widely understood tools [19, 24, 29, 42, 81].
7. Auditors should be provided with validation datasets, including fault-seeded inputs and edge cases, to enable testing and verification of audit tools [34, 48, 74].
8. Where feasible, verifier software should be derived from formally verified specifications to improve assurance in its correctness [31–33].
9. Third-party implementations should rely on transparent development processes, ensure long-term availability of verifier software, and account for operational constraints of the election context [48].

Incentives for independent auditing. The third group concerns mechanisms for encouraging independent participation:

10. A culture of independent auditing should be fostered throughout the development and deployment of the e-voting system [7].
11. Incentives should be provided to encourage third parties to perform audits [35, 68]. For example, financial compensation or formal recognition mechanisms may be used to motivate participation [66].

Discussion

E-voting systems may fail to support meaningful independent audits if potential auditors cannot realistically implement, validate, or operate verifier software. This highlights a fundamental practical limitation: auditability depends not only on the availability of audit mechanisms, but on their accessibility and usability by independent parties.

A key tension arises between facilitating independent audits and protecting sensitive information. Enabling independent verification requires the disclosure of election artefacts and system details, which may be restricted by legal or regulatory constraints. When such constraints limit access to audit data, they reduce the pool of authorised auditors and thereby weaken the practical independence of the audit process.

As a result, trust becomes concentrated in a limited set of privileged actors who are permitted to access the necessary information. This introduces a trade-off between transparency and regulatory compliance, where stronger privacy or legal guarantees may come at the cost of reduced audit independence.

Recommendations

We derive the following recommendations for enabling independent audits and third-party verifier implementations:

6A Provide independent auditors access to all non-confidential audit artefacts, technical specifications, and verifier tooling necessary to conduct meaningful verification.

6B Provide sufficient time before the election for independent auditors to implement, validate, and test verifier software.

6C Publish verifier software, technical specifications, and audit artefacts in publicly accessible and understandable formats to support independent implementation.

6D Reduce technical barriers to independent auditing by favouring simple verifier designs, readable specifications, and widely used, standardised algorithms where possible.

6E Provide complete validation datasets, including edge cases and fault-seeded inputs, to enable third parties to test and verify their implementations.

6F Encourage sustained third-party auditing through transparent development processes and appropriate incentives.

4.2 Verifier specification development framework

The verifier specification development framework is a six-step process that considers system design and the election context to derive an election audit methodology and a verifier specification that supports third-party implementation. The framework takes as input the protocol design, claimed security properties, and trust assumptions. Its output is a verifier specification containing verification requirements, required audit data, audit methodology, and documentation supporting independent audits and verifier implementation. The development framework is presented in Figure 4.1.

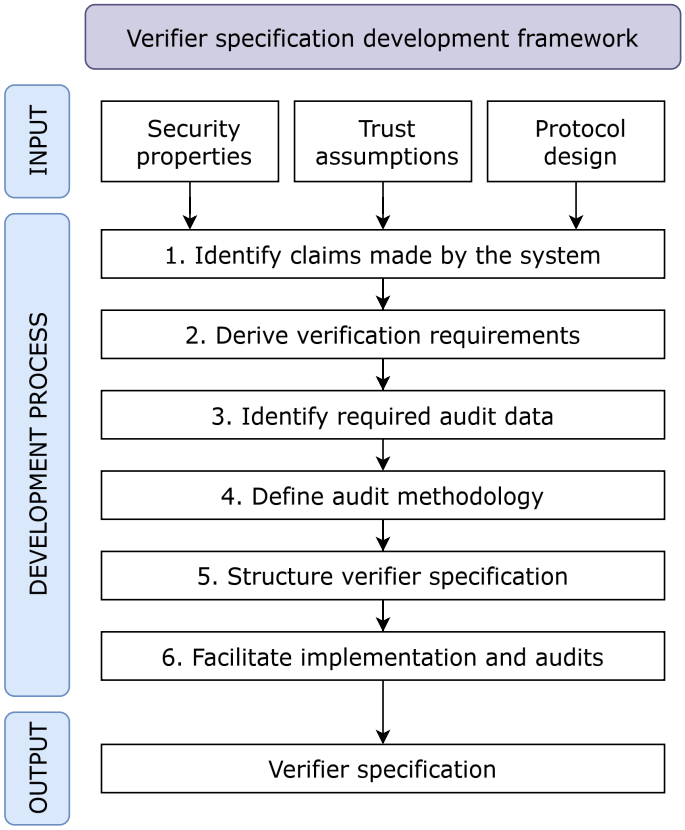

Figure 4.1: Process model of the verifier specification development framework

The process can be divided into two stages: deriving verification requirements and audit methodology (steps 1–4), and structuring the verifier specification and facilitating independent audits (steps 5–6). Each framework step is based on recommendations identified during the literature review, presented in Section 4.1. To use the framework, the steps are performed sequentially, following the supporting recommendations noted in Table 4.2. The purpose of each framework step is as follows:

1. **Identify claims made by the system** by reviewing the protocol design, claimed security properties, and trust assumptions.
2. **Derive verification requirements** by identifying what must be verified during an election audit.
3. **Identify required audit data** by determining which audit artefacts are necessary to evaluate the verification requirements.
4. **Define audit methodology** by defining how each verification requirement is evaluated during the election audit.
5. **Structure verifier specification** by organising the system claims, verification requirements, required audit data, audit methodology, and verification tests into a verifier specification.
6. **Facilitate implementation and audits** by identifying supporting artefacts, validation data, documentation, and other measures needed by independent auditors and verifier developers.

No.	Framework step	Supporting recommendations
1	Identify claims	4A, 5A
2	Derive requirements	1B
3	Identify audit data	2A, 2B
4	Define audit methodology	3A, 3B, 3C, 3D, 4C, 5C, 6D
5	Structure specification	5B, 5E, 5F
6	Facilitate audits	1A, 3E, 3F, 4B, 5D, 6A, 6B, 6C, 6E, 6F

Table 4.2: Supporting recommendations for each framework step.

4.3 Case study

The case study follows the methodology outlined in Section 3.3. First, the current and revised Swiss Post e-voting system protocols are presented, where differences between these systems are highlighted. Then, we apply the verifier specification development framework to each difference to determine the impact of each change on the verification process. To determine the exact impact on the current verifier specification and audit methodology, these are also described.

4.3.1 The Swiss Post e-voting system

The Swiss Post e-voting system is a verifiable e-voting system developed by the Swiss Post. It allows voters to cast ballots remotely, using personal devices, in national and cantonal elections in Switzerland [69]. As an e-voting system intended for national elections in Switzerland, it is regulated by the VEleS [20].

This sub-section presents the current and revised versions of the Swiss Post e-voting system. The two system versions are presented together, implying similarity between them, except when differences are explicitly highlighted.

The presentation of the Swiss Post e-voting system contains information previously presented in the pre-project [64]. The paragraphs have since been revised to varying degrees.

Election context

Although the Swiss Post is an e-voting system vendor, they are not an election authority running elections. The Swiss Post is a system vendor that provides the e-voting system to cantons, which in turn use it to set up and run their own elections[3] [20, Art. 14].

As mentioned above, the VEleS imposes requirements on e-voting systems to be used in national and cantonal elections in Switzerland [20]. Therefore, several design choices are governed by legislation rather than a decision by the Swiss Post.

Claimed security properties

The Swiss Post e-voting system is required by the VEleS to provide a series of security objectives. These have been grouped following the desired security properties presented in Section 2.2.2:

1. **Privacy:**
   - Preserve vote privacy and exclude premature partial results [20, Art. 4].
   - Protect personal information relating to voters [20, Art. 4].
   - Ensure that evidence of voting behaviour is not maliciously exploited [20, Art. 4].
2. **Integrity:**
   - Ensure the accuracy of the result [20, Art. 4].
3. **Robustness:**
   - Ensure the soundness of generated proofs [20, Art. 6].
   - Protect information intended for voters from manipulation [20, Art. 4].
   - Ensure availability and operability of the voting system [20, Art. 4].
4. **Verifiability:**
   - Provide individual verifiability [20, Art. 5].

[3] https://swisspost-digital.ch/en/solutions/e-voting/success-through-cooperation

   - Provide universal verifiability [20, Art. 5].
5. **Fairness and Eligibility:**
   - A set of authentication credentials cannot be used to cast more than one vote [20, Art. 2, Annex 2.12.1].
   - Each ballot must be cast using a set of authentication credentials established during the setup phase [20, Art. 2].

Most of these claimed security properties are defined as presented in Section 2.2.2, but the VEleS has specific requirements regarding universal verifiability. For universal verifiability to hold, the evidence examined during an audit has to verify three properties regarding the votes tallied to reach the election result:

1. *All votes cast in conformity with the system that have been registered by the trustworthy part of the system* [20, Art. 5.3.a].
2. *Only votes cast in conformity with the system* [20, Art. 5.3.a].
3. *All partial votes in accordance with the proof generated in the individual verification process* [20, Art. 5.3.a].

These points concern the integrity of the tally phase. The first point states that there must be proof that all correctly cast votes were included in the tally, demonstrating that no correctly cast votes were rejected. The second point states that there must be proof that no non-conforming votes are present in the tally. The third point states that there must be proof that all partial votes verified in the individual verification process were used in the tally.

An additional requirement from Swiss law regarding universal verifiability mandates that the auditors must evaluate the evidence from the protocol execution in an observable procedure, where technical aids must be isolated and independent of the rest of the e-voting system [20, Art. 5.3.b].

Protocol design

The Swiss Post e-voting system allows voters to cast their ballots remotely using a private device. It employs return codes to provide individual verifiability and ensure vote privacy through a mix-net [71]. The system is developed by the Swiss Post, and several design requirements are defined by legislation [20].

The system is highly complex, and the presentation will therefore cover the system protocols only to the extent necessary to perform the evaluation in Section 4.3.2. The presentation of the system will follow the three phases of an e-voting system, starting with the setup phase:

Setup phase. The purpose of the setup phase is to set up the infrastructure components and instantiate cryptographic variables to be used later during the election process. A critical part of the Swiss Post e-voting system is a voting card, which contain login codes and the codes required for individual verifiability. The voting cards and the

codes on them are created during this phase, printed by a single printing component, and distributed to each voter by post [14, 71]. The main difference between the two system versions lies in how voting cards are created:

Current system: A single Setup Component is responsible for creating the return codes required for individual verifiability and for setting up the voting cards. Each code required for individual verifiability is split into four shares and distributed so that each of the four return code Control Component infrastructure participants receives one share of each code [71, p. 60].

Revised system: The creation of the voting cards is a split task between four Online Setup Components and a single Offline Setup Component. As with the current system, the return codes required for individual verifiability are split into four, with each Online Control Component holding on to one share of each code. The process is executed to ensure that no single component knows the complete return code on the voting cards. The Offline Setup Component is disconnected from the system following the setup phase. It remains offline until the tally phase, preventing it from performing any actions during the voting phase [14].

Voting phase. To cast votes, voters use their personal user devices to access a web page, called the voting client, which interacts with the voting server. The voting cards each user received during the setup phase contain a unique login code for authentication and a set of codes required for return-code-based individual verifiability. Each voting option has a corresponding return code, while each voting card has a unique approve code and finalisation code [14, 71]. We have found that the current and revised system versions use different names for these codes, but for simplicity, we have chosen to use only those proposed by the new system [14].

After being authenticated, the voters cast and confirm their votes through two protocols: SendVote and ConfirmVote. The voter performs these through their voting client. They choose a voting option and cast their vote using the SendVote protocol. As the system uses return codes for individual verifiability, a return code must be provided to the voter. Four Online Control Components each provide a share of the return code corresponding to the voter and the specific voting option. Combined, these shares constitute a return code that the voter then compares with the return code for their intended voting option, as shown on their voting card. This step ensures that the ballot received by the voting server contained their intended voting option [71, p. 89] [14].

The voter then uses the ConfirmVote protocol to confirm that the return codes were as expected, indicating that the vote recorded by the voting server was correct, and that their cast ballot should be added to the ballot box. To initiate the protocol, the voter provides the approve code, as shown on their voting card. The voting server

then records the ballot in the ballot box, before finishing the protocol by providing the voter with a finalisation code. To verify that the vote was recorded as cast, the voter must compare the finalisation code received with the finalisation code on their voting card [71, p. 99] [14].

The infrastructure participants responsible for providing the return code and finalisation code under these two protocols are called return code Control Components in the current system version and Online Control Components in the revised version. These are deployed in a group of four, with each component holding one share of each code, requiring all components to provide code shares that are combined to construct the correct code [71, pp. 89, 99] [14]. We identify a small difference regarding which system participant is responsible for combining the code shares into a full code:

Current system: The voting server is responsible for combining the code shares into a full code, which is then sent to the voting client [71, pp. 89, 99].

Revised system: The voting client communicates directly with each Online Control Component and receives the code shares from each of them. The voting client is therefore responsible for combining the code shares [14].

Tally phase. The tally phase begins with anonymising the ballots using a Bayer-Groth mix-net [5], which employs the verifiable mechanisms of malleable commitments and verifiable mixing [70, p. 35], described in Section 2.2.3. Four mixing components, called Online Control Components, sequentially shuffle the ballots and perform partial decryptions on the recorded votes. Each of these steps generates evidence, in the form of proof of shuffling and decryption [71, p. 105]. After the shuffling and partial decryption by the Online Control Components, the Tally Control Component verifies the proofs generated by the voting clients and the prior proofs of correct shuffling and decryption [71, pp. 120, 121]. Given that these proofs are valid, the Tally Control Component performs another shuffle before the final decryption of the votes [71, p. 122]. The result is a list of plaintext votes, which are processed to produce a tally file containing the final tally [71, pp. 123–124].

Trust assumptions

The trust assumptions of the Swiss Post e-voting system are presented using the same groupings as in Section 2.3.

Cryptographic assumptions. As presented in Section 2.3, the security of a cryptographic system is derived by reducing the problem to a known hard problem. In Switzerland, the hard problems must be generally accepted and recognised cryptographic methods [20, Annex 2.14.3]. The current and revised versions have slight differences in which hard problems they trust:

Current system: For the current Swiss Post e-voting system, the hard problems are decisional Diffie-Hellman (DDH) and subgroup-generated-by-small-primes (SGSP) [70, p. 55].

Revised system: The newly proposed protocol removes the SGSP trust assumption by changing the group relied on to a generic group [14], following a planned measure [21, A.13] to remove reliance on the SGSP trust assumption. The change also allows for elliptic curves to be used instead of finite fields [14].

Software assumptions. To ensure the software's authenticity for voters and Cantons, the Federal Chancellery requires a trusted build and deployment process [20, technical requirement for e-voting 24.3]. This concerns the voting client that each voter interacts with (both the frontend and backend software) and the setup component that each Canton uses to configure their elections [73].

To meet the requirement of a trusted build process, Swiss Post provides auditors with the ability to reproduce the build process through the open source code and compare the hash of their compilation artefacts to the published hash checksums [73]. This enables auditors to verify that the published hash checksums are from the correct source code and software. The trusted deployment process involves auditors who verify that the software used for the Swiss Post frontend and backend, and for the Cantonal infrastructure, matches the published verified hash checksums [73]. Through these steps, the assumption of correct and trusted software has been replaced with trust in the build and deployment process.

The verifier specification and source code are publicly available[4], allowing third parties to audit and implement the verifier using the verifier specification [72]. Furthermore, a bug bounty programme[5] is used to motivate the public into studying the system and verifier specifications, and submit identified bugs through a bug reporting portal[6]. This means that the soundness of both the system and the verifier rests on the assumption that independent auditors both detect and report any potential issues.

Infrastructure and operation assumptions. The Swiss Federal Chancellery has evaluated each permitted communication link for trustworthiness [20, Annex 2.10]. In Section 2.3, a general trust assumption was presented regarding the link between the voter and the user device, which can be trusted for privacy [25, p. 504]. This assumption is also permitted in the Swiss context, as well as a similar trust assumption for the links between the auditors and their technical aids, the setup component and the auditors' technical aids, and the link between the printing component and the voters/voting cards [20, Annex 2.10.2]. In the Swiss Post e-voting system, the

[4] https://gitlab.com/swisspost-evoting/
[5] https://www.post.ch/en/about-us/responsibility/swiss-post-bug-bounty
[6] https://yeswehack.com/programs/swiss-post

trust placed in the postal service, as the link between the printing component and the voters, cannot be changed [14].

The system participants in the Swiss Post e-voting system are connected to many trust assumptions for the proofs that have to be provided for individual verifiability [20, Annex 2.5], universal verifiability [20, Annex 2.6], authentication [20, Annex 2.8], and voting privacy and exclusion of premature partial results from the final tally (premature result) [20, Annex 2.7]:

- The printing component has to be trusted for almost every property: Authentication [20, Annex 2.9.4.2], vote privacy and exclusion of premature results [20, Annex 2.9.3.2], and for the soundness of the proof of individual verifiability [20, Annex 2.9.1.2].
- The mix-net control components have a weaker trust assumption where only one of four needs to be trusted for all the proofs [20, Annex 2.9.1.2, 2.9.2.2, 2.9.3.2, 2.9.4.2].
- Only one auditor and one technical aid per trusted auditor has to be trusted for the soundness of the proof of universal verifiability [20, Annex 2.9.4.2].
- User devices used by the voters can only be trusted for vote privacy, and we must trust that no adversaries have control of user devices for the exclusion of premature results. [20, Annex 2.9.3.2].

For determining the soundness of the proof of individual verifiability of the e-voting system [20, Annex 2.5], the Federal Chancellery allows an assumption of trustworthiness for the setup component, print component and one of the four mix-net control components [20, Annex 2.9.1.2].

When determining the soundness of the proof of universal verifiability [20, Annex 2.6], the trust assumptions are again one of the four mix-net control components, but also one auditor of the election, and one technical aid from the auditor [20, Annex 2.9.2.2].

In summary, the trust assumptions for the current and revised system versions are similar. However, the trust assumptions diverge regarding the setup phase, following a planned measure [21, A.5] to remove the strong trust assumption in the setup component:

Current system: The setup component has to be trusted for all the same properties as the printing component: Authentication [20, Annex 2.9.4.2], vote privacy and exclusion of premature results [20, Annex 2.9.3.2], and for the soundness of the proof of individual verifiability [20, Annex 2.9.1.2].

Revised system: The new proposal for the setup phase of the Swiss Post e-voting system restructures the cryptographic protocol to use five setup components with a weaker trust assumption: four Online Setup Components and one Offline Setup Component. Only one of the Online Setup Components has to be trusted

for the authentication and individual verifiability. For vote privacy, only one of the five setup components has to be honest and trusted [14]. However, both the link between the printing component and the voter, and the printing component itself are still considered trustworthy [14].

Current election audit methodology

The Swiss Post provides the source code for the verifier software and a verifier specification[7], as required by Swiss law. [20, Art. 11–12]. Together with other documentation on the system, available on their GitLab page[8], these artefacts provide auditors with insight into how the system works and its audit methodology, and enable third parties to both audit the verifier software itself and compile the source code to perform election audits.

Ahead of an election, the published documentation and source code have to be audited. The VEleS requires that the Federal Chancellery appoint such auditors to audit these public artefacts [20, Art. 10], but the public should also be provided with the ability to report issues [20, Art. 13]. Members of the public who provide suggestions that improve the security of the system are to be financially compensated by the cantons [20, Art. 13.3]. Although such audits are distinct from election audits, they serve to verify the software used during an election and the election audit.

The current election audit methodology for the Swiss Post e-voting system consists of a first set of verifications following the setup phase and a second set following the tally phase [72]. The audit is performed by a single or multiple auditors, using at least one technical aid, identified as the verifier [72].

The algorithm *VerifyConfigPhase* is executed by the verifier to verify the setup phase [72, p. 16], while *VerifyTally* is executed to verify the correctness of the tally phase [72, p. 30]. Each algorithm verifies that the audit data is as expected:

- To ensure the completeness of the audit data, the verifier verifies that the audit data contains all expected items, is in the correct file path, and is in the proper format [72, pp. 16, 30].
- The authenticity of the datasets by verifying the digital signatures [72, pp. 17, 31].
- The consistency between elements is verified through verifying that the encryption group parameters, file names, IDs, proofs, keys, encrypted votes and number of votes are consistent [72, pp. 20–24, 34–37].

Neither the *VerifyConfigPhase* nor the *VerifyTally* performs any verification solely to verify the integrity of the data. Instead, the integrity of the elements in the dataset can be considered verified if none of the other verifications rejects their input [72, pp. 25, 37].

[7] https://gitlab.com/swisspost-evoting/verifier/verifier
[8] https://gitlab.com/swisspost-evoting/

However, the two algorithms necessarily differ regarding verifying evidence:

VerifyConfigPhase. The evidence from the setup phase is used by the VerifyConfigPhase algorithm to verify that the setup phase resulted in cryptographically sound parameters, the correct number of voting cards, that each voting option is assigned its own prime number, and that all relevant configuration information was correctly signed. These verifications indicate to the auditors whether the results from the setup phase will lead to reasonably secure election operations [72, p. 25].

VerifyTally. The verification of the tally phase uses the presented evidence to verify several properties, including the accuracy of the tally and the well-formedness of the ballots, ensuring that universal verifiability in the Swiss context holds [72, p. 38]:

- The proofs generated by the voting clients when the voters cast their ballots are verified. When the verifier performs this verification, it is itself verified to use the same mapping of primes as the Online Control Components.
- The Online Control Components' proofs of correct shuffle and decryption are verified.
- The proofs generated by the Tally Control Component while performing the shuffle, final decryption, processing of the plaintext votes, and the generation of the tally file are verified.

4.3.2 Evaluation of the system revision

The evaluation of the current and revised versions of the Swiss Post e-voting system is conducted using the verifier specification development framework presented in Section 4.2. The evaluation is structured according to the sequence of framework steps.

For each supporting recommendation (Rec.) under each framework step, we denote the relevant system change from the current to the revised versions, as described in Section 4.3.1. The evaluation analyses how each identified protocol change affects the resulting verifier specification, including its verification requirements, required audit artefacts, verification procedures, and documentation. Particular focus is placed on whether the verifier specification requires modification or the addition of something new, such as additional audit artefacts or entirely new verification procedures. The symbol "–" denotes that no relevant change was identified for a recommendation and that no significant impact on the verifier specification was identified.

Identify claims made by the system

The first step of the framework is to identify and document the claimed security properties and to define the context in which these are to be supported. As presented in Table 4.3, the primary changes for Recommendations 4A and 5A concern the trust assumptions in the setup phase, given the new distributed setup procedures.

Rec.	Relevant system change	Impact on verifier
4A	Modified trust assumptions	Modify: trust model
5A	Modified trust assumptions	Modify: trust model

Table 4.3: Evaluation of the *identify claims made by the system* step.

While the overall security objectives remain unchanged, the trust model by which these are achieved has changed. Therefore, the verifier specification requires updated documentation of the trust assumptions associated with the setup phase.

Derive verification requirements

Rec.	Relevant system change	Impact on verifier
1B.a	New proofs generated	New: verification methods
1B.b	New proofs generated	Modify: verification methods
1B.c	New proofs generated	Modify: verification methods
1B.d	New proofs generated	Modify: verification methods
1B.e	New proofs generated	Modify: verification methods

Table 4.4: Evaluation of the *derive verification requirements* step.

The evaluation of the verification-requirement step, presented in Table 4.4, indicates that the revised protocol introduces additional evidence verification requirements due to the generation of new cryptographic proofs. The introduction of distributed setup procedures generates new cryptographic proofs, requiring both new evidence-verification procedures and modifications to existing audit data-verification procedures.

Identify required audit data

Rec.	Relevant system change	Impact on verifier
2A	New proofs generated	Modify: provided audit data
2B	Modified protocol design	Modify: system documentation

Table 4.5: Evaluation of the *identify required audit data* step.

The changes to the protocol design introduce new proofs from the distributed setup phase and the reorganisation of responsibilities during the voting phase. This change necessitates modifying the audit data and its documentation, as presented in Table 4.5. These proofs must be added to the audit data provided to auditors and to the overview of available audit data.

Define audit methodology

Rec.	Relevant system change	Impact on verifier
3A	New proofs from setup phase	New: verification methods
3B	–	–
3C	–	–
3D	New verification methods	New: describe degrees of failure
4C	New trust assumptions	Modify: ensure consistency with trust model
5C	New verification methods	Modify: audit documentation
6D	New verification methods	New: low complexity documentation

Table 4.6: Evaluation of the *define audit methodology* step.

How the system revision impacts the framework step for defining the audit methodology is presented in Table 4.6. The new proofs generated by the new protocol design require the introduction of new verification methods. These verification methods must then be documented in an accessible manner that includes the possible degrees of failure for the test, in accordance with Recommendation 6D.

Structure verifier specification

Rec.	Relevant system change	Impact on verifier
5B	Modified protocol design	Modify: system documentation
5E	Modified protocol design	New: pseudocode algorithms
5F	–	–

Table 4.7: Evaluation of the *structure verifier specification* step.

The fifth step of the framework is to structure and organise the inputs and the results from the previous steps into a cohesive document. The identified impact of the system revision on this step is presented in Table 4.7. The modification to the system protocol design requires updates and additions to the system documentation to describe the new protocol clearly. The new algorithms introduced by the revised protocol should be represented through pseudocode for readability.

Facilitate implementation and audits

The final step of the framework is to facilitate independent audits of the specification and third-party implementations of the verifier software. As presented in Table 4.8, most of the supporting recommendations were not impacted by the system revision. Recommendation 1A requires that the new verifier be verified, and Recommendation 5D and Recommendation 6C require that the published documentation and the

Rec.	Relevant system change	Impact on verifier
1A	Modified protocol design	New: verify the new verifier
3E	–	–
3F	–	–
4B	–	–
5D	Modified protocol design	Modify: system documentation
6A	–	–
6B	–	–
6C	New documentation and verifier	Modify: update published artefacts
6E	New verification methods	New: validation datasets
6F	–	–

Table 4.8: Evaluation of the *facilitate implementation and audits* step.

verifier source code be updated to accurately reflect the system changes. However, Recommendation 6E necessitates the creation of new validation datasets for third parties to verify their own, or the Swiss Post's, verifier software.

Summary of identified impacts

Framework step	Stability across versions	Main impact type
Identify claims	Weakly affected	Trust model updates
Derive requirements	Moderately affected	New verification logic
Identify audit data	Moderately affected	Additional artefacts
Define audit methodology	Strongly affected	New procedures
Structure specification	Moderately affected	Documentation updates
Facilitate implementation	Weakly affected	New verifier & documentation

Table 4.9: Summary of the impact the system revision had on each framework step.

To summarise the impacts of the identified system changes on the results produced by each framework step, we have constructed Table 4.9. The stability across system revisions has been evaluated as either weakly affected, indicating a small impact on the outcome; moderately affected, indicating a medium impact on the outcome; or strongly affected, indicating that the two system versions lead to a distinctly different outcome. The table also notes the main impact type.

The evaluation of this section has identified the extent to which the different framework steps are affected by the revised Swiss Post e-voting system, introducing distributed setup proofs and additional audit artefacts. The main impact on the framework concerns the audit methodology, while the security claims and the facili-

tation of independent audits and third-party verifier implementations remain largely intact.

Chapter 5
Discussion

This chapter discusses the results presented in Chapter 4. The aim is to interpret the main findings, relate them to the literature, and assess the extent to which they answer the research questions.

The discussion is organised into three sections. Section 5.1 discusses the literature review and its role in forming the basis for the proposed framework. Section 5.2 discusses the verifier specification development framework in relation to RQ1 and RQ2. Section 5.3 discusses the Swiss Post case study in relation to RQ3 and RQ4.

The chapter focuses on the findings most relevant to evaluating the framework and understanding its limitations, rather than discussing all results in equal detail.

5.1 Literature review

The aim of the literature review was to consolidate recommendations for conducting election audits of universally verifiable e-voting systems. Through an intentionally broad query, 60 works contributed to the derivation of 25 recommendations across six aspects of the election audit process.

The identified recommendations show that achieving universal verifiability in e-voting systems depends not only on protocol design, but also on practical aspects of auditability, including documentation quality, the availability of audit data, and support for independent verification. This indicates that systems must enable independent parties to verify election outcomes in practice, rather than relying only on the theoretical verifiability properties of the underlying protocol. In particular, practical verifiability requires auditability and implementability to be considered during system design and documentation.

A central finding of the review is that existing research provides complementary but fragmented guidance. Prior work proposes audit frameworks, verification methods, and implementation practices, but few contributions address how these aspects should be combined to support end-to-end, independently verifiable elections. By consolidating these perspectives, the literature review provides the basis for the

verifier specification development framework presented in Section 4.2.

The analysis also highlights several tensions relevant to the development of verifier specifications. First, there is a trade-off between transparency and regulatory or operational constraints, as independent audits require access to artefacts that may be restricted by law or by practical considerations. Second, there is a tension between system complexity and auditability. Simplified documentation can improve accessibility for auditors and verifier developers, but may introduce inaccuracies or omit details needed for implementation. Conversely, complete and precise documentation can support correctness, but may require substantial expertise to interpret and maintain.

These tensions show that practical verifiability depends on more than cryptographic mechanisms. The usability, accessibility, and maintainability of audit artefacts and verification tools also influence whether independent auditing is feasible. High complexity and specialised expertise requirements may limit the number of parties capable of performing meaningful audits, even when the necessary documentation and audit data are publicly available.

The limited degree of disagreement identified in the literature suggests that the recommendations are broadly aligned with existing work. However, consensus is not proof that all relevant considerations have been identified. Underlying assumptions common within the field of universally verifiable elections may have gone undetected. Furthermore, although the query was intended to be broad, it did not extensively cover related fields such as formal verification of cryptographic protocols or standardisation processes. Including these fields might have provided additional insights into assumptions and methods relevant to verifier specifications.

Consequently, the review may reproduce common perspectives present in the literature while overlooking considerations that have received less attention. The recommendations should therefore be understood as a structured synthesis of existing guidance, rather than as a complete set of requirements for all universally verifiable e-voting systems.

Overall, the literature review supports the need for a framework that connects protocol design, audit data, verification procedures, documentation, and implementability. The recommendations provide a foundation for such a framework, but their completeness and practical usefulness must be assessed through the framework development and case study discussed in the following sections.

5.2 Verifier specification development framework

From the recommendations identified during the literature review, we derived a development framework for verifier specifications. The framework provides a higher-level methodology for integrating these recommendations into a coherent process for developing verifier specifications. By employing the development framework, system vendors are provided with guidance to support the development of verifier

specifications for independent audits and third-party implementation of verifier software.

As noted in Section 1.5, a consolidated list of recommendations for universally verifiable election audits has not been defined previously. The literature review suggests that existing recommendations are largely complementary rather than contradictory. However, this does not mean that they are directly applicable as a development method. The recommendations identify properties, artefacts, and practices relevant to election audits, but they do not define how these should be ordered, combined, or translated into a verifier specification. Through the verifier specification development framework, we therefore refine the usability of the recommendations by organising them into a process for developing verifier specifications and facilitating independent audits.

In themselves, the categories defined during the literature review do not indicate the sequential order of applying each recommendation, neither between categories nor within them. This is exemplified by the trust assumptions category, which concerns assumptions that may affect several stages of both verifier development and election audits. It is also illustrated by the verification requirements category, where dependencies exist between recommendations. Recommendation 1A concerns verification of the verifier software and e-voting system, while Recommendation 1B concerns verification of an election event. If Recommendation 1B is followed without first addressing Recommendation 1A, the audit may rely on an unsound verifier. In that case, the verifier would be unable to provide meaningful guarantees about the election event. Therefore, restructuring the recommendations into a distinct framework is necessary to make them more accessible during the development of verifier specifications.

Each framework step is based on one or more recommendations identified during the literature review. This provides traceability between the framework structure and the literature from which it was derived. As a result, the reason for including each step can be inspected and reassessed if the underlying body of literature changes.

This traceability is useful if individual recommendations are later revised, replaced, or found to be unsupported. In that case, the affected framework step can be identified and reconsidered. However, larger changes in the underlying literature may also require changes to the framework's structure.

The following subsections discuss the framework in relation to RQ1 and RQ2. Section 5.2.1 discusses whether the framework supports the derivation of verification requirements from system claims and trust assumptions. Section 5.2.2 discusses whether the resulting requirements can be organised into a verifier specification that supports independent audits and third-party implementation.

5.2.1 RQ1 – Deriving verification requirements

RQ1: *How can high-level security properties and trust assumptions be systematically translated into explicit, measurable, and testable verification requirements for verifiable voting protocols?*

The process of deriving verification requirements is defined primarily by the first two steps of the verifier specification development framework: *identify claims made by the system* and *derive verification requirements*. However, these requirements are made actionable through the third and fourth steps: *identify required audit data* and *define audit methodology*. Therefore, all four steps must be discussed to assess whether the framework derives verification requirements that are explicit, measurable, and testable.

Identify claims made by the system

The goal of the first step is to identify and formalise the system's claims, given the protocol design, trust assumptions, and claimed security properties. This step is necessary because the audit process cannot directly verify abstract properties such as universal verifiability or voter privacy. Instead, these properties must be expressed as claims about what the system provides and under which assumptions it is provided.

Through the supporting recommendations 4A and 5A, the first framework step creates traceability between the system claims and their basis in the protocol design, trust model, and stated security properties. This limits the risk that verification requirements are derived without a clear basis. However, this depends on the method used to achieve the relevant security objectives and on the trust assumptions being identifiable from the available system documentation.

Derive verification requirements

The second step derives verification requirements from the identified claims, supported by recommendation 1B. These requirements are divided between requirements that verify evidence produced during protocol execution and requirements that verify the audit data itself, including completeness, integrity, authenticity, and consistency.

This division is important because an auditor must verify not only that the individual proofs are valid, but must also verify the audit data itself. If the audit dataset is incomplete, inconsistent, or not authentic, the audit process may be unsound even if the individual tests are correct. Therefore, the framework requires that the verification process consider both the evidence and the audit data on which it depends.

The categorisation of verification requirements derived through the literature review aligns with the categories presented in Section 2.4.2, which were drawn from Haenni et al. [28]. This alignment should not be understood as the prior categorisation

determining the review's results. Rather, the recommendation categories were derived from the reviewed literature, while the terminology used to describe them was adopted from Haenni et al., where the concepts overlapped. The convergence suggests that the resulting categories are relevant to the election audit process, but it does not prove that all relevant categories have been identified.

Identify required audit data and define audit methodology

Steps three and four make the derived verification requirements measurable and testable. Step three identifies the audit data required to evaluate each requirement, while step four defines how the evaluation is performed through the audit methodology. Together, these steps connect each requirement to the artefacts needed for verification, the procedure used to evaluate them, and the conditions under which the requirement is satisfied or rejected.

A verification requirement becomes measurable when a condition is defined, along with the relevant audit data and the criteria for success or failure. It becomes testable when the audit methodology specifies how to evaluate this condition. Without these steps, the framework would identify what should be verified but provide no actionable method for verifying it. Consequently, steps three and four form the link between the derivation of verification requirements and their use in an election audit.

Summary and limitations

Together, the first four framework steps address RQ1 by providing a structured method for translating security objectives and trust assumptions into verification requirements. The requirements are explicit because they are derived from identified system claims. They are measurable because they are defined as conditions linked to specific audit data, with defined criteria for success and failure. They are testable because the audit methodology defines how each requirement is evaluated.

However, two main limitations remain. First, the derivation process depends on the competence of the verifier specification developer and on the quality of the available system documentation. If relevant claims, trust assumptions, or protocol details are missing or misunderstood, this may propagate to the derived verification requirements. The framework may help make such gaps visible, but it cannot, by itself, correct missing or inaccurate system documentation.

Second, the framework does not assess whether the protocol's evidence-generation model is sufficient for verifying all identified claims. Step three may reveal that required evidence is missing, but resolving this issue would require changes to the system protocol rather than only changes to the verifier specification. Therefore, when applying the framework, it is necessary to assume that the protocol produces sufficient evidence to verify the claimed properties.

To address RQ1 directly, the framework provides a systematic method for deriving explicit, measurable, and testable verification requirements for verifiable e-voting protocols, given high-level security properties, trust assumptions, and protocol documentation. However, the resulting requirements depend on the quality of these inputs and on the sufficiency of the protocol's evidence-generation model. The framework should therefore be understood as a structured derivation method, rather than as a guarantee that all relevant verification requirements have been identified.

5.2.2 RQ2 – Structuring the verifier specification

RQ2: *How can these verification requirements be organised into a systematic, modular, and implementation-agnostic verifier specification that supports independent development and evaluation?*

RQ2 concerns the second stage of the verifier specification development framework. While RQ1 concerns how verification requirements are derived, RQ2 concerns how these requirements are organised into a verifier specification and how the specification can support independent use. In this context, independent evaluation primarily refers to independent election audits but also includes independent reviews of the verifier specification and the verification process. This is addressed primarily through step five, *structure verifier specification*, and step six, *facilitate implementation and audits*.

Structure verifier specification

The main function of step five is to organise the results from the previous framework steps into a verifier specification. This includes system claims, verification requirements, required audit data, verification procedures, and the test catalogue. However, not all documentation recommendations are assigned to step five. Several are connected to earlier framework steps because the relevant information should be documented as soon as it is identified. For example, system claims and trust assumptions are documented during step one, while audit data and verification procedures are documented during steps three and four.

This distribution of documentation recommendations is important because traceability becomes difficult to reconstruct after the development process is complete. By documenting the output of each step as it is produced, the risk of omitting relevant results from the verifier specification is reduced. Furthermore, this approach creates a structure in which the outputs of each framework step correspond to specific sections of the verifier specification.

This emergent structure is important because it makes the verifier specification easier to inspect. A verifier specification that only lists tests may describe what the verifier should do, but not why the tests are necessary or how they support the

claimed security properties. By preserving the links between claims, requirements, audit data, and procedures, the framework clarifies the role of each test.

The framework also supports modularity by organising verification tests according to their corresponding verification requirements. This allows the verification process to be divided into smaller parts that can be inspected, implemented, and updated separately. However, this modularity depends on the verifier specification maintaining clear boundaries between requirements and tests.

The framework supports implementation-agnostic specifications by recommending that verification procedures and algorithms be documented in pseudocode. This can guide verifier software developers without requiring them to follow a specific software implementation. However, the degree of implementation independence is limited by the underlying e-voting protocol. If the protocol depends on specific data formats, cryptographic libraries, or implementation choices, the verifier specification may be only partially implementation-agnostic.

Facilitate implementation and audits

Step six addresses the practical conditions required to facilitate the development of independent verifiers and independent election audits. This step is based on recommendations from several categories, indicating that independence is not achieved only by structuring the verifier specification. It also depends on the publication of supporting artefacts, validation data, documentation, and on providing independent parties with sufficient time to review the verification process before an election event.

The framework supports independent election audits by requiring the verifier specification to document the audit methodology, required audit data, verification procedures, and reporting process. These elements allow auditors to understand what is verified, which artefacts are needed, how each verification test is performed, and how the results should be interpreted. Without this information, auditors may be able to run verifier software, but they would have limited ability to assess whether the verification process itself is complete and sound.

This distinction is important because independent audits require more than access to verifier software. If auditors only receive the output of a vendor-provided verifier, they remain dependent on the vendor's implementation and interpretation of the election audit process. A verifier specification reduces this dependence by making the verification process auditable. An independent audit of the verifier specification can then assess whether the test catalogue covers the relevant verification requirements, whether the required audit data is available, and whether the verification report supports the election audit's conclusion.

The framework also supports third-party verifier development, but this is a related and more implementation-focused goal. Publishing documentation or source code alone does not necessarily enable the development of independent verifiers. Third-

party developers also need clear documentation to support implementation, including pseudocode, input formats, and validation datasets for both successful and failed verifications. These artefacts help developers understand the required verification logic and validate that their implementation behaves correctly.

These measures support independent audits by making the verification process auditable and support third-party verifier development by providing information needed to implement and validate verifier software. However, both uses depend on practical conditions outside the framework itself, including access to relevant artefacts, validation data, documentation, sufficient time, and competent independent parties. These limitations are discussed further below.

Summary and limitations

Together, steps five and six address RQ2 by organising the derived verification requirements into a verifier specification and by identifying supporting measures for independent verifier development and independent election audits. The specification is systematic because it preserves the relationship between system claims, verification requirements, audit data, and verification procedures. It is modular because tests are organised according to verification requirements. It is partly implementation-agnostic because verification procedures may be described in pseudocode rather than as source code for a specific implementation.

However, these properties are conditional. The degree of implementation independence depends on the underlying protocol design and on whether the specification separates verification logic from implementation-specific details. Independent verifier development and independent election audits also depend on the availability of supporting artefacts, validation data, and documentation, as well as sufficient time before the election event. If relevant artefacts are confidential, legally restricted, or unavailable for operational reasons, independent auditors may be unable to fully evaluate the election event, even if the verifier specification is well structured.

The framework also does not remove the need for specialised expertise. Independent auditors must be able to assess the verifier specification, interpret the verification report, and evaluate whether the audit conclusion is supported by the evidence. Verifier software developers may receive more concrete guidance through pseudocode and validation data, but they still require sufficient technical competence to implement the verification procedures correctly. Therefore, the framework can reduce reliance on vendor-provided verifiers and opaque audit procedures, but it does not eliminate the need for competent independent auditors and developers.

To address RQ2 directly, the framework provides a structured method for organising verification requirements into a modular verifier specification that can support independent election audits and third-party verifier development. However, it should be understood as providing support for these goals, rather than ensuring that third

parties can implement verifier software or perform independent audits in practice.

5.2.3 Limitations of framework

Omissions or areas not covered in the literature review may affect the derivation of the framework's steps and inputs. Although the review employed an intentionally broad query string and snowballing, it is possible that relevant recommendations or contrary opinions were not identified. As a consequence, the framework should be interpreted as reflecting the current body of literature rather than representing a definitive or exhaustive model for verifier specification development.

Although the framework is based on the results of a structured literature review, its sequential structure reflects an interpretation of the relationships among the identified recommendations. While some dependencies between recommendations and development steps are apparent, the literature generally describes desired properties of the audit process and e-voting systems, rather than defined development process steps. Therefore, alternative framework structures may also satisfy the identified recommendations. The proposed sequence should therefore be viewed as one plausible representation of the literature rather than the only possible option.

A key assumption for the framework's usability is that the verifier specification developer can identify the relevant system claims and trust assumptions and has a firm grasp of the system protocol's inner workings. These items constitute the inputs of the framework, and therefore the foundation of the derived verifier specification. If these are incomplete or incorrect, for example, due to insufficient system documentation, this may propagate into the resulting specification.

The final step of the framework introduces an additional safeguard by requiring that the verifier specification itself, and any verifier software developed in accordance with it, be audited, preferably by an independent auditor. This may help identify errors introduced earlier in the process, including errors resulting from poor system documentation or incorrect interpretation of the protocol. However, this does not guarantee that all deficiencies in the inputs are detected or resolved. Consequently, the framework should be understood as supporting a structured development process for verifier specifications, not as guaranteeing that the resulting specification is complete or correct.

5.3 Case study

As presented in Section 3.3, the purpose of the case study is not primarily to evaluate the Swiss Post e-voting system itself, but to evaluate the verifier specification development framework when applied to two related e-voting protocols within the same election context. The availability of both the current and revised Swiss Post system designs provides an opportunity to assess, through RQ3, whether the

framework appears independent of a specific protocol design, and, through RQ4, whether it can identify and capture audit-relevant differences between protocol designs.

A methodological strength of the case study is that the security objectives used as input to the framework remained largely unchanged across the two system versions, as they are primarily determined by legislation. Therefore, the differences identified by the framework can mainly be attributed to changes in protocol design and trust distribution, rather than to changes in the objectives of the revised system. This provides a useful basis for evaluating the framework, as the case study compares two protocol designs against a mostly stable set of security objectives and election context.

5.3.1 RQ3 – General applicability of framework

RQ3: *How do variations in protocol design, trust distribution, and evidence-generation mechanisms influence both the derived verification requirements and the resulting structure of the verifier specification?*

The objective of RQ3 is to assess whether the framework depends on specific e-voting architectures or can be applied across systems with different protocol designs, trust models, and evidence-generation mechanisms. The revised Swiss Post protocol provides a useful basis for this discussion because it introduces a new protocol design for the setup phase, necessitated by a new trust model, while the election context and security objectives remain the same.

Where RQ4 concerns whether the framework captures concrete differences between the current and revised Swiss Post protocols, RQ3 concerns the more general question of whether such differences require changes to the framework itself.

Dependence on protocol design

The case study indicates that the framework is not tied to the specific protocol design of the current Swiss Post system. Although the revised Swiss Post protocol introduces changes to the setup and voting phases, all six framework steps remain applicable. The development process described by the framework is therefore not changed by the revised protocol design.

Instead, the protocol changes primarily affect the outputs of the framework steps. Changes from the current to the revised protocol may introduce new verification requirements, audit artefacts, and verification procedures, while the overall framework structure remains unchanged. However, because the case study compares only two related protocol designs within the same election context, it supports only a limited claim of protocol-design independence.

Dependence on evidence-generation mechanisms

The case study forms a more limited basis for assessing the framework's dependence on evidence-generation mechanisms. The revised Swiss Post protocol changes parts of the protocol design and trust distribution, but it does not introduce a fundamentally different mechanism for providing universal verifiability or verifiable vote privacy. Therefore, the case study cannot by itself establish that the framework is independent of alternative evidence-generation mechanisms.

Nevertheless, the case study suggests that the framework is independent of protocol design, indicating that different evidence-generation mechanisms would primarily affect the content of the verifier specification rather than the framework itself. Alternative methods of achieving verifiability, such as those presented in Section 2.2.3, may change which evidence must be collected, validated, and linked to security claims. However, they do not remove the need to identify system claims, derive verification requirements, define audit data, specify audit procedures, and document the resulting verification process.

This suggests that the framework is structurally applicable across different evidence-generation mechanisms. However, the audit methodology derived from the framework remains dependent on the specific evidence produced by the protocol. Therefore, the framework may remain applicable even if the resulting verifier specification differs substantially.

Dependence on trust distribution

The revised Swiss Post protocol also modifies the distribution of trust among system participants, introducing a new context in which the security objectives must be met. The framework appears robust to such changes because trust assumptions are treated as explicit input. Therefore, changes to the trust model affect the results of the framework steps rather than the framework structure itself. As identified in the literature review, trust assumptions limit what can be verified, since assumed properties are not directly supported by audit evidence. Introducing additional trust assumptions may therefore reduce what must be verified, because more properties are assumed rather than demonstrated. Conversely, removing trust assumptions may require additional verification tests, because properties that were previously trusted must instead be supported by audit evidence.

This behaviour is a methodological strength of the framework. Rather than embedding a particular trust model in the development process, the framework uses the trust model as input and provides a process for assessing its impact on the audit process. However, the case study evaluates this feature only within a single system context, so further studies would be required to determine whether the same holds for systems with substantially different trust models.

Summary

Source of variation	Effect on framework outputs	Effect on framework structure
Protocol design	Changes verification requirements, audit artefacts, and verification procedures.	Framework steps remain applicable in the Swiss Post case.
Trust distribution	Changes which properties are assumed and which must be supported by audit evidence.	Trust assumptions are treated as framework input.
Evidence-generation mechanisms	Changes which evidence must be collected, validated, and linked to system claims.	Not fully evaluated in the case study; conclusion remains tentative.

Table 5.1: Summary of how protocol variation affects framework outputs and structure.

Table 5.1 summarises the main findings for RQ3. Overall, the case study indicates that variations in protocol design and trust distribution primarily affect the framework's outputs rather than its structure. The framework steps remained applicable to both versions of the Swiss Post system, while the resulting verification requirements, required audit data, and audit methodology changed in line with the revised protocol design and trust model.

For evidence-generation mechanisms, the conclusion is more tentative. The case study does not include a system revision using a fundamentally different verifiability mechanism. However, the framework distinguishes between the general process of developing verifier specifications and the specific evidence produced by a protocol. Therefore, RQ3 is partly answered: the case study supports the view that the framework is structurally applicable across related protocol designs and trust models, but further cases would be required to assess its general applicability across substantially different e-voting systems.

5.3.2 RQ4 – Applicability during system revision

RQ4 *How effectively does the proposed derivation and structuring method capture and differentiate verification requirements when applied to both the current and revised Swiss Post e-voting system designs?*

The objective of RQ4 is to evaluate whether the framework produces different audit-relevant outputs when applied to different protocol designs. In contrast to RQ3, which concerns whether the framework structure remains applicable under variation, RQ4 concerns whether the framework captures the consequences of concrete changes in the input. The changes between the two Swiss Post e-voting system versions are summarised in Table 5.2, which forms the basis for discussing whether the framework identifies differences in verification tasks and in the resulting verifier specification.

In this discussion, *capturing differences* means identifying how protocol changes affect the verification process, while *differentiating verification requirements* means determining whether the changes require new, modified, removed, or unchanged verification methods to verify the verification requirements.

Protocol Change	Observed impact on verification tasks
New cryptographic group	Cryptographic primitives used during verification must be updated.
New trust model	Reduction in trust increases the number of proofs verified as evidence.
New setup and voting phase	Introduction of new system participants increase the number of operations needing to be verified.

Table 5.2: Summary of protocol changes and their impact on verification tasks.

Ability to capture differences

The revised protocol introduces three main categories of changes, as presented in Table 5.2. Although these changes do not alter the security objectives, they affect how these objectives are supported by evidence and verified during the audit process. The case study indicates that the framework captures these effects by identifying changes to verification requirements, required audit data, and verification procedures.

This is most evident in how the framework connects protocol changes to audit consequences. A change in the cryptographic group affects the cryptographic primitives used during verification. A change in the distribution of trust affects which properties must be verified rather than assumed. Changes to the setup and voting phases affect which protocol operations and system participants must be included in the audit process. These differences show that the framework does not produce identical verification tasks when the input protocol changes.

At the same time, the structure of the verification tasks remains largely unchanged. This is expected because the framework organises tests according to the verification requirements they support. Since the high-level security objectives remain the same across the two Swiss Post versions, the main structure of the verification requirements also remains stable. The differences are therefore primarily found in the contents of the verifier specification, rather than in its overall organisation.

This stability should not be interpreted as a failure to differentiate between the two protocol designs. Rather, it suggests that the framework separates the structure of the verifier specification from the protocol-specific content placed within that structure. This is useful during system revision, because it allows verifier specification developers to compare versions within a stable organisational model while still identifying which verification tasks, artefacts, and procedures must be changed.

Ability to differentiate required changes

The findings indicate that although the framework can capture the effects of specific protocol changes, its use for updating an existing verifier specification is more limited. The framework can identify how a change affects the verifier specification that should result from the revised protocol. However, it does not, by itself, determine whether an existing verifier specification is sufficient or whether specific verification methods must be added, modified, or removed.

Therefore, using the framework during system revision requires an additional step of comparison. First, the differences between the current and revised systems must be identified. Second, each framework step must be applied to determine which supporting recommendations are affected by the identified differences. Third, the resulting impacts must be compared with the existing verifier specification to determine whether they require new verification methods, modifications to existing methods, removal of obsolete methods, or no change.

Although this process is not simple, it still provides guidance during system revision. The framework helps structure the analysis of how protocol changes affect verification tasks, and can direct verifier specification developers towards the parts of the verifier specification most likely to require updates. Therefore, the framework supports differentiation between required changes, but does not automate the revision of an existing verifier specification.

Summary

To address RQ4 directly, the case study indicates that the framework can capture audit-relevant differences between the current and revised Swiss Post e-voting protocols. The framework produced different outputs where the protocol input changed, including changes to verification tasks, audit artefacts, and verification procedures. This suggests that the framework can support the maintenance and revision of verifier specifications when e-voting systems evolve.

However, the framework does not automatically identify all necessary changes to an existing verifier specification. Its usefulness during revision depends on comparing the framework outputs with the existing verifier specification and on the verifier specification developer's ability to interpret the consequences of protocol changes. Therefore, the framework makes the revision process more structured, but not automatic.

5.3.3 Limitations of case study

The findings of the case study are constrained by several limitations. First, the case study is based on a single e-voting system. Although the current and revised Swiss Post systems provide a useful comparison, they remain two versions of the

same system within the same election context. Therefore, the case study provides limited support for claims about the framework's generalisability across different e-voting systems. Additional case studies using systems with different verifiability mechanisms, trust models, and documentation practices would be required to assess whether the framework is applicable beyond this case.

Another limitation concerns the treatment of the verifier specification in the thesis. The framework treats the verifier specification as a single conceptual artefact containing the information needed by verifier software developers and election auditors. In practice, this information may be distributed across several documents. In the Swiss Post system, for example, information relevant for understanding the election audit process is found in the verifier specification [72], while information relevant for understanding the protocol steps is found in the system specification [71]. This distribution may reduce duplication and maintenance effort, but it also increases the effort required for third parties to understand the complete verification process. As a result, applying the framework may require considering the complete documentation set rather than the verifier specification alone.

A further limitation is that the case study does not produce a complete revised verifier specification. Instead, it focuses on whether the framework captures the audit consequences of the revised protocol design. As a result, the case study can identify which verification requirements, audit artefacts, and verification procedures are affected by the revision, but it does not assess whether a complete verifier specification developed through the framework would be correct, complete, or independently implementable.

The case study has also not been evaluated through independent implementation of verifier software or through use by external auditors. Therefore, it should be viewed as an initial evaluation of the framework's applicability, rather than a complete validation of its practical usefulness. The case study evaluates whether the framework can be applied to a real e-voting system and whether it captures differences between two related protocol designs. However, it does not demonstrate that independent parties can, in practice, use the framework to develop verifier specifications, implement verifier software, or perform election audits.

Chapter 6
Conclusion

This thesis has investigated how verifier specifications for universally verifiable e-voting systems can be developed in a systematic manner to support independent election audits and third-party implementation of verifier software.

6.1 Main contributions

The main contribution of this thesis is the verifier specification development framework presented in Section 4.2. The framework is based on recommendations derived from the current state of research on universally verifiable e-voting systems, election audits, verifier specifications, and implementability.

The framework provides a structured method for deriving and organising verifier specifications for verifiable e-voting systems. It connects system claims, trust assumptions, verification requirements, required audit data, audit methodology, and documentation. Through this structure, the framework supports translating high-level security properties and trust assumptions into explicit, measurable, and testable verification requirements.

The discussion in Sections 5.2 and 5.3 indicates that the framework can support the development of verifier specifications by preserving traceability between system claims, verification requirements, audit artefacts, and verification procedures. The case study further indicates that the framework can capture audit-relevant consequences of protocol changes. When applied to the current and revised Swiss Post e-voting protocols, the framework identified changes to verification requirements, audit artefacts, and verification procedures resulting from changes in protocol design and trust distribution.

Furthermore, the case study highlights the impact of the revised Swiss Post e-voting protocol on the existing verification procedure and, by extension, its verifier specification. These results may provide guidance on which parts of the verifier specification to focus on when revising the audit process for the updated protocol.

These findings should be interpreted in light of the thesis's limitations. The

framework reflects the literature included in the review, and alternative framework structures may also satisfy the identified recommendations. Furthermore, the case study is limited to two versions of the Swiss Post e-voting system within the same election context. Consequently, the thesis supports the framework's applicability in this setting, but provides limited evidence of its general applicability.

Although the findings indicate that the framework can support independent audits and the development of third-party verifiers, they do not demonstrate this in practice. As a result, the findings only suggest this ability without evidence that independent parties can use it to develop verifier specifications, implement verifier software, or perform election audits in practice.

6.2 Future work

The most important direction for future work is to evaluate the framework through practical use by independent verifier specification developers, verifier software developers, and election auditors. Such a study would provide stronger evidence of whether the framework is understandable, useful, and sufficiently complete for practical use. It would also help determine whether the framework reduces the effort required to develop verifier specifications and supports independent election audits in practice.

The research also highlighted the importance of usability in the election audit process. While this thesis focused on the development of verifier specifications, the usability of verifier software and verification reports is also integral to a sound election audit. Therefore, future work should investigate how verifier specifications, verifier software, and verification reports can be designed to reduce the expertise required for auditors while still preserving the precision needed for a sound verification process. In addition, further research should investigate incentives for independent auditors and third-party verifier developers, as the availability of documentation alone does not ensure independent participation.

The findings indicate that third parties' ability to conduct audits may be constrained by legal and operational requirements. Future work should therefore investigate how transparency requirements can be balanced against restrictions on the publication of audit artefacts, system documentation, and other information relevant to the verification process.

Although the use of formal verification for verifier software development was introduced in Section 2.4.1 and identified in the literature review, the potential impact of this development methodology on the proposed development framework was not extensively covered in this thesis. The field of formal verification may provide insight not gathered during the literature review. Future work should therefore examine the potential impact of formal verification on the audit process and whether it imposes requirements on the verifier specification beyond those identified in this thesis.

Recommendations for conducting election audits of universally verifiable elections may change as new research emerges. Future research should therefore update the literature review to determine whether any identified recommendations have become obsolete, whether new recommendations are needed, or whether existing recommendations should be revised.

References

1. Antonyan, T., Davtyan, S., *et al.*: State-Wide Elections, Optical Scan Voting Systems, and the Pursuit of Integrity. IEEE Transactions on Information Forensics and Security **4**(4), 597–610 (2009)
2. Arafat, S.M.: On the Estonian Internet Voting System, IVXV, SoK and Suggestions, (2025). https://eprint.iacr.org/2025/506. Cryptology ePrint Archive, Paper 2025/506.
3. Babenko, L., Pisarev, I.: E-Voting System Based on Multiple Ballot Casting. In: Fourth International Congress on Information and Communication Technology, pp. 89–97 (2020)
4. Basin, D., Radomirovic, S., Schmid, L.: Alethea: A provably secure random sample voting protocol. In: 2018 IEEE 31st Computer Security Foundations Symposium, pp. 283–297 (2018)
5. Bayer, S., Groth, J.: Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012, pp. 263–280. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
6. Benaloh, J., Naehrig, M., Pereira, O.: REACTIVE: Rethinking Effective Approaches Concerning Trustees in Verifiable Elections. In: Electronic Voting, pp. 17–37. Springer Nature Switzerland (2026)
7. Buckland, R., Teague, V., Wen, R.: Towards best practice for e-election systems: Lessons from trial and error in Australian elections. In: E-Voting and Identity, pp. 224–241. Springer Berlin Heidelberg (2012)
8. Cansell, D., Gibson, J.P., Méry, D.: Refinement: A Constructive Approach to Formal Software Design for a Secure e-voting Interface. In: Proceedings of the First International Workshop on Formal Methods for Interactive Systems, pp. 39–55 (2007)
9. Chang-Fong, N., Essex, A.: The Cloudier Side of Cryptographic End-to-end Verifiable Voting: A Security Analysis of Helios. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 324–335 (2016)
10. Chaum, D., Ryan, P., Schneider, S.: A practical voter-verifiable election scheme. In: Computer Security – ESORICS 2005, p. 139 (2005)
11. Chondros, N., Zhang, B., *et al.*: D-DEMOS: A Distributed, End-to-end Verifiable, Internet Voting system. In: 2016 IEEE 36th International Conference on Distributed Computing Systems, pp. 711–720 (2016)

12. Clarkson, M., Chong, S., *et al.*: Civitas: Toward a secure voting system. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy, pp. 354–368 (2008)

13. Cortier, V., Debant, A., Gaudry, P.: Breaking Verifiability and Vote Privacy in CHVote. In: Computer Security – ESORICS 2025, pp. 86–105 (2026)

14. Cortier, V., Debant, A., *et al.*: A Practical and Fully Distributed E-Voting Protocol for the Swiss Context, Cryptology ePrint Archive, Paper 2025/1625 (2025). https://eprint.iacr.org/2025/1625.

15. Cyrus, M., Tiwari, M.: Formally Verified Verifiable Group Generators. In: Fundamentals of Software Engineering. Lecture Notes in Computer Science, p. 186 (2025)

16. Driza Maurer, A.: The swiss post/Scytl transparency exercise and its possible impact on internet voting regulation. In: E-VOTE ID-2019. Lecture Notes in Computer Science, pp. 83–99 (2019)

17. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)

18. Erdmanis, J.: Unconditional Individual Verifiability with Receipt Freeness via Post-Cast Isolation, Cryptology ePrint Archive, Paper 2025/1186 (2025). https://eprint.iacr.org/2025/1186 (last visited: Apr. 14, 2026).

19. Essex, A., Clark, J., *et al.*: Eperio: Mitigating Technical Complexity in Cryptographic Election Verification. In: Electronic Voting Technology – Workshop on Trustworthy Elections (2012). https://eprint.iacr.org/2012/178 (last visited: Apr. 14, 2026)

20. Federal Chancellery, Federal Chancellery Ordinance on Electronic Voting (2022)

21. Federal Chancellery, Vote électronique: Catalogue of measures by the Confederation and cantons (2023)

22. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86, pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)

23. Garera, S., Rubin, A.: An Independent Audit Framework for Software Dependent Voting Systems. In: CCS'07: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 256–265 (2007). https://dl.acm.org/doi/pdf/10.1145/1315245.1315278

24. Gawel, D., Kosarzecki, M., *et al.*: Apollo - End-to-End Verifiable Internet Voting with Recovery from Vote Manipulation. In: ELECTRONIC VOTING, E-VOTE-ID 2016, pp. 125–143 (2017)

25. Gjøsteen, K.: Practical Mathematical Cryptography. Chapman and Hall/CRC, New York (2022)

26. Gjøsteen, K.: The Norwegian Internet Voting Protocol, Cryptology ePrint Archive, Paper 2013/473 (2013). https://eprint.iacr.org/2013/473.

27. Haenni, R., Dubuis, E., *et al.*: CHVote: Sixteen Best Practices and Lessons Learned. In: Electronic Voting (E-Vote-ID 2020), pp. 95–111. Springer International Publishing (2020). https://link.springer.com/10.1007/978-3-030-60347-2_7 (last visited: Apr. 20, 2026)

28. Haenni, R., Dubuis, E., *et al.*: Process Models for Universally Verifiable Elections. In: Electronic Voting (E-Vote-ID 2018), pp. 84–99. Springer International Publishing (2018)

29. Haenni, R., König, R.E., *et al.*: CHVote Protocol Specification, Cryptology ePrint Archive, Paper 2017/325 (2017). https://eprint.iacr.org/2017/325 (last visited: Apr. 15, 2026).

30. Haines, T., Lewis, S., *et al.*: How not to prove your election outcome. In: 2020 IEEE Symposium on Security and Privacy, pp. 644–660 (2020)

31. Haines, T., Goré, R., Stodart, J.: Machine-Checking the Universal Verifiability of ElectionGuard. In: Secure IT Systems (NordSec 2020), pp. 57–73 (2021). https://link.springer.com/chapter/10.1007/978-3-030-70852-8_4

32. Haines, T., Goré, R., Sharma, B.: Did you mix me? Formally Verifying Verifiable Mix Nets in Electronic Voting. In: 2021 IEEE Symposium on Security and Privacy, pp. 1748–1765. IEEE (2021). https://ieeexplore.ieee.org/document/9519460/ (last visited: Apr. 20, 2026)

33. Haines, T., Goré, R., Tiwari, M.: Verified Verifiers for Verifying Elections. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 685–702. ACM (2019). https://dl.acm.org/doi/10.1145/3319535.3354247 (last visited: Apr. 20, 2026)

34. Haines, T., Rønne, P.: New Standards for E-Voting Systems: Reflections on Source Code Examinations. In: Financial Cryptography and Data Security. FC 2021 International Workshops, pp. 279–289. Springer Berlin Heidelberg (2021). https://link.springer.com/10.1007/978-3-662-63958-0_24 (last visited: Apr. 20, 2026)

35. Haines, T., Rose, J.: Vestigial Vulnerabilities in Deployed Verifiable E-Voting Systems, Cryptology ePrint Archive, Paper 2025/2033 (2025). https://eprint.iacr.org/2025/2033 (last visited: Apr. 14, 2026).

36. Halderman, J., Teague, V.: The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election. In: E-VOTING AND IDENTITY, VOTEID 2015, pp. 35–53 (2015)

37. Heiberg, S., Martens, T., *et al.*: Improving the Verifiability of the Estonian Internet Voting Scheme. In: Electronic Voting, pp. 92–107. Springer International Publishing (2017)

38. Jafar, U., Ab Aziz, M., Shukur, Z.: Blockchain for Electronic Voting System-Review and Open Research Challenges. SENSORS **21**(17) (2021)

39. Jivanyan, A., Feickert, A.: Aura: private voting with reduced trust on tallying authorities, Cryptology ePrint Archive, Paper 2022/543 (2022). https://eprint.iacr.org/2022/543 (last visited: Apr. 14, 2026).

40. Kiayias, A., Michel, L., *et al.*: Tampering with special purpose trusted computing devices: A case study in optical scan e-voting. In: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 30–39 (2007)

41. Kremer, S., Ryan, M., Smyth, B.: Election Verifiability in Electronic Voting Protocols. In: Computer Security – ESORICS 2010, pp. 389–404. Springer Berlin Heidelberg (2010)

42. Kulyk, O., Volkamer, M.: Usability is not Enough: Lessons Learned from 'Human Factors in Security' Research for Verifiability. In: Third International Joint Conference on Electronic Voting (E-Vote-ID 2018) (2018). https://eprint.iacr.org/2018/683 (last visited: Apr. 14, 2026)

43. Kumar, A., Choudhary, M., *et al.*: Permissioned Smart Contract Based e-Voting System for University. In: 2025 International Conference on Networks and Cryptology (NETCRYPT), pp. 1339–1343 (2025)

44. Küsters, R., Müller, J.: Cryptographic Security Analysis of E-voting Systems: Achievements, Misconceptions, and Limitations. In: Electronic Voting (E-VOTE-ID 2017), pp. 21–41 (2017)

45. Lundin, D., Ryan, P.: Human readable paper verification of prêt à voter. In: Computer Security - ESORIC 2008, Proceedings, p. 395 (2008)

46. Martínek, T., Pelikán, M., *et al.*: Implementation of a Secret and Verifiable Personal Remote Electronic Election of an Agrarian Organization per the Recommendation of the Council of Europe. AGRIS ON-LINE PAPERS IN ECONOMICS AND INFORMATICS **16**(3), 59–73 (2024)

47. McMurtry, E., Pereira, O., Teague, V.: When Is a Test Not a Proof? In: Computer Security – ESORICS 2020, pp. 23–41. Springer International Publishing (2020). https://link.springer.com/10.1007/978-3-030-59013-0_2 (last visited: Apr. 20, 2026)

48. Moser, F., Grimm, R., *et al.*: Recommendations for Implementing IV in Internet Voting. In: E-Vote-ID 2024, pp. 35–53. Gesellschaft für Informatik (2024). https://dl.gi.de/server/api/core/bitstreams/a9e478ac-5f21-4d91-ac08-1d9d39c38834/content (last visited: Apr. 20, 2026)

49. Moser, F., Kirsten, M., Dörre, F.: SoK - Mechanisms Used in Practice for Verifiable Internet Voting. In: E-Vote-ID 2024, pp. 141–162. Gesellschaft für Informatik, Bonn (2024)

50. Moser, F., Müller, J., *et al.*: A Study of Mechanisms for End-to-End Verifiable Online Voting. Tech. rep., Bundesamt für Sicherheit in der Informationstechnik (2024). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Cryptography/End-to-End-Verifiable_Online-Voting.pdf?___blob=publicationFile&v=4

51. Neumann, S., Olembo, M., *et al.*: Helios verification: To alleviate, or to nominate: Is that the question, or shall we have both? In: Electronic Government and the Information Systems Perspective (EGOVIS 2014), p. 260 (2014)

52. Oechsner, S., Pereira, V., Scholl, P.: Who Verifies the Verifiers? Lessons Learned From Formally Verified Line-Point Zero-Knowledge. IACR Communications in Cryptology **2**(3) (2025)

53. Panja, S., Roy, B.K.: A secure end-to-end verifiable e-voting system using zero knowledge based blockchain, Cryptology ePrint Archive, Paper 2018/466 (2018). https://eprint.iacr.org/2018/466 (last visited: Apr. 14, 2026).

54. Perez, A.J., Ceesay, E.N.: Improving End-to-End Verifiable Voting Systems with Blockchain Technologies. In: 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data, pp. 1108–1115 (2018)

55. Rivest, R.L.: On the notion of 'software independence' in voting systems. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **366**(1881), 3759–3767 (2008)

56. Rønne, P., Ryan, P., Smyth, B.: Cast-as-Intended: A Formal Definition and Case Studies. In: Financial Cryptography and Data Security, FC 2021, pp. 251–262 (2021)

57. Ryan, P.: Prêt à Voter with Paillier encryption. Mathematical and Computer Modelling **48**(9), 1646–1662 (2008). https://www.sciencedirect.com/science/article/pii/S0895717708001763

58. Ryan, P.: The computer ate my vote. Formal Methods: State of the Art and New Directions (2010). https://www.scopus.com/pages/publications/84892227222?origin=resultslist

59. Ryan, P., Teague, V.: Pretty good democracy. In: Security Protocols XVII (Security Protocols 2009), pp. 111–130 (2013). https://www.scopus.com/pages/publications/84872435183?origin=resultslist

60. Ryan, P., Bismark, D., *et al.*: Pret a Voter: a Voter-Verifiable Voting System. IEEE Transactions on Information Forensics and Security **4**(4), 662–673 (2009)

61. Shahandashti, S., Hao, F.: DRE-ip: A verifiable e-voting scheme without tallying authorities. In: Computer Security - ESORICS 2016, PT II, p. 240 (2016). https://www.scopus.com/inward/record.uri?eid=2-s2.0-84990864132&doi=10.1007%2f978-3-319-45741-3_12&partnerID=40&md5=d6ce4cf01de25cfd72073aa7bc320fba

62. Al-Shammari, A.F.N., Villafiorita, A., Weldemariam, K.: Understanding the Development Trends of Electronic Voting Systems. In: 2012 Seventh International Conference on Availability, Reliability and Security, pp. 186–195 (2012)

63. Sherman, A.T., Carback III, R.T., *et al.*: Scantegrity, en-US. https://cisa.umbc.edu/research/scantegrity/.

64. Strømsnes, A.F.: Verifying the Swiss Post e-voting system, ed. by T. Silde and A. Høgåsen. Unpublished pre-project for this thesis (2025).

65. Sturton, C., Jha, S., *et al.*: On Voting Machine Design for Verification and Testability. In: CCS'09: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 463–476 (2009). https://people.eecs.berkeley.edu/~daw/papers/vvm-ccs09.pdf

66. Sudharsan, B., Tharun, V.R., *et al.*: Secured Electronic Voting System Using the Concepts of Blockchain. In: 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0675–0681 (2019)

67. Suharsono, T.N., Gunawan, *et al.*: Univer Metric to Measure the Degree of Universal Verifiability in E-Voting. In: 2024 18th International Conference on Telecommunication Systems, Services, and Applications (TSSA), pp. 1–4. IEEE (2024). https://ieeexplore.ieee.org/document/10863897/ (last visited: Apr. 20, 2026)

68. Sutopo, A., Haines, T., Rønne, P.: On the Auditability of the Estonian IVXV System And an Attack on Individual Verifiability. In: Financial Cryptography and Data Security. FC 2023 International Workshops, pp. 19–33 (2024)

69. Swiss Post, E-voting Online voting and elections, en. https://swisspost-digital.ch/en/solutions/e-voting.

70. Swiss Post, Protocol of the Swiss Post Voting System: Computational Proof of Complete Verifiability and Privacy. Tech. rep., version 1.4.0. Technical report. Swiss Post (2025). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/Protocol/Swiss_Post_Voting_Protocol_Computational_proof.pdf

71. Swiss Post, Swiss Post Voting System: System Specification. Tech. rep., version 1.5.2. Technical report. Swiss Post (2025). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/System/System_Specification.pdf

72. Swiss Post, Swiss Post Voting System: Verifier Specification. Tech. rep., version 1.6.0. Technical report. Swiss Post (2025). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/System/Verifier_Specification.pdf

73. Swiss Post, Trusted Build and Trusted Deployment of the Swiss Post Voting System. Tech. rep., Technical report. Swiss Post (2025). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/master/Trusted-Build/Trusted%20Build%20of%20the%20Swiss%20Post%20Voting%20System.md

74. Treier, T.: Beyond the Happy Path: A Safety-Critical Audit Methodology for Estonia's I-Voting Processing Application. IEEE Access **13**, 203429–203443 (2025)

75. Treier, T., Duuna, K.: Identifying and Solving a Vulnerability in the Estonian Internet Voting Process: Subverting Ballot Integrity Without Detection. IEEE Access **12**, 197766–197782 (2024)

76. United Nations, Goal 10, https://sdgs.un.org/goals/goal10#targets_and_indicators.

77. United Nations, Goal 16, https://sdgs.un.org/goals/goal16#targets_and_indicators.

78. Valmised, https://www.valimised.ee/en/internet-voting-estonia.

79. Weldemariam, K., Kemmerer, R., Villafiorita, A.: Formal analysis of an electronic voting system: An experience report. Journal of Systems and Software **84**(10), 1618–1637 (2011)

80. Xia, Z., Tong, Z., *et al.*: Framework for practical and receipt-free remote voting. IET Information Security **12**(4), 326–331 (2018)

81. Yang, Y., Guan, Z., *et al.*: PriScore: Blockchain-Based Self-Tallying Election System Supporting Score Voting. IEEE Transactions on Information Forensics and Security **16**, 4705–4720 (2021)

NTNU

Kunnskap for en bedre verden