TAL TECH

DOCTORAL THESIS

Testing for Trust: Safety-Critical Auditability and Limits of Coercion Resistance in the Estonian Internet Voting System

Tarvo Treier

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2026

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
49/2026

Testing for Trust: Safety-Critical Auditability and Limits of Coercion Resistance in the Estonian Internet Voting System

TARVO TREIER

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer Science on 7 July 2026

Supervisor: Assoc. Prof. Innar Liiv,
Department of Software Science, School of Information Technologies,
Tallinn University of Technology
Tallinn, Estonia

Opponents: Dr. Peter Roenne,
University of Luxembourg,
Esch-sur-Alzette, Luxembourg

Dr. Jan Willemson,
Cybernetica AS,
Tartu, Estonia

Defence of the thesis: 1 September 2026, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Tarvo Treier _____
signature

ISSN 2585-6898 (paperback)
ISBN 978-9916-80-527-5 (paperback)
ISSN 2585-6901 (PDF)
ISBN 978-9916-80-527-5 (PDF)
https://doi.org/10.23658/taltech.49/2026
Printed by EVG Print

Treier, T. (2026). *Testing for Trust: Safety-Critical Auditability and Limits of Coercion Resistance in the Estonian Internet Voting System* [TalTech Press]. https://doi.org/10.23658/taltech.49/2026

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
49/2026

Usaldus läbi testimise: ohutuskriitiline auditeeritavus ja sundimiskindluse piirid Eesti internetihääletussüsteemis

TARVO TREIER

TAL TECH
KIRJASTUS

Contents

List of Publications 7

Author's Contributions to the Publications 8

Abbreviations 9

Terms 10

Symbols 13

1 Introduction 14
- 1.1 Problem Statement and Motivation 14
- 1.2 Research Objectives and Approach 16
- 1.3 Structure of the Dissertation 18

2 Background 19
- 2.1 Assurance concepts and evidence in i-voting 19
- 2.2 IVXV and the processing stage as an evidential bottleneck 20
- 2.3 Testing for trust as safety-critical audit validation 22
- 2.4 Revoting secrecy under composition with the eID ecosystem 22

3 Results 24
- 3.1 A checkable integrity invariant for the processing stage (Publication I) 24
  - 3.1.1 Auditability gap and contribution 24
  - 3.1.2 Artefact model and the invariant 24
  - 3.1.3 Detection capability, boundary conditions, and practical uptake 25
- 3.2 Safety-critical auditability of the processing application (Publication II) 25
  - 3.2.1 From "auditing the system" to "testing the audit" 25
  - 3.2.2 Requirements, fault scenarios, and explicit oracles 26
  - 3.2.3 Phase A coverage and sensitivity snapshot 26
  - 3.2.4 Interpreting the instructive miss and the mapping presentation 26
- 3.3 Revoting secrecy under composition with eID transaction logs (Publication III) 27
  - 3.3.1 Threat model and the leakage channel 27
  - 3.3.2 Mitigation strategies and comparative assessment 28
  - 3.3.3 Implications for interpreting processing-stage complexity 28

4 Discussion 29
- 4.1 From isolated checks to an assurance view 29
  - 4.1.1 An assurance trajectory across the three publications 29
  - 4.1.2 A compact claim–evidence–boundary synthesis 29
  - 4.1.3 Why an integrity oracle is necessary but not sufficient 30
- 4.2 Testing for trust: conceptual implications 31
  - 4.2.1 Trust as a conclusion rather than a premise 31
  - 4.2.2 An asymmetry between correctness assurance and privacy assurance 31
  - 4.2.3 Verifiability evidence is broader than cryptographic proofs 31
  - 4.2.4 Evidence surfaces, explicit oracles, and semantic constraints 32
- 4.3 Implications for audit practice and system governance 33
  - 4.3.1 Pre-election validation of audit toolchains 33

4.3.2 Who audits matters as much as how auditing is done ... 33
4.3.3 From pass/fail outcomes to sensitivity and coverage reporting ... 33
4.3.4 Normative clarity and implicit policy discovered in code ... 33
4.3.5 Governance implications of the eID logging ecosystem ... 34
4.4 Generalisability and significance ... 34
4.4.1 What generalises are patterns rather than IVXV artefacts ... 35
4.4.2 A policy fork and its effect on the technical design space ... 35
4.5 Limitations and future work ... 35
4.5.1 Limitations ... 35
4.5.2 Future work ... 36

5 Conclusion ... 37
5.1 Answers to the research questions ... 37
5.1.1 RQ1: How can processing-stage integrity be verified from artefacts without trusting the internal logic of the processing application? ... 37
5.1.2 RQ2: How can safety-critical testing principles be applied to evaluate the quality and sensitivity of auditing tools? ... 37
5.1.3 RQ3: How do eID transaction logs affect revoting secrecy and coercion-mitigation assumptions? ... 37
5.2 Implications for practice and governance ... 38
5.2.1 Pre-election validation using fault-seeded corpora ... 38
5.2.2 Stating tool scope and evidence surfaces explicitly ... 38
5.2.3 Composition-aware interpretation of coercion mitigation ... 38
5.3 Closing remarks ... 38

List of Figures ... 40

List of Tables ... 41

References ... 42

Acknowledgements ... 45

Abstract ... 46

Kokkuvõte ... 48

Appendix 1 ... 51

Appendix 2 ... 71

Appendix 3 ... 89

Appendix 4 ... 109

Curriculum Vitae ... 114

Elulookirjeldus ... 116

List of Publications

The present Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers. A full list of the author's other publications is provided in the curriculum vitae at the end of the dissertation.

I T. Treier and K. Düüna. Identifying and solving a vulnerability in the Estonian internet voting process: Subverting ballot integrity without detection. *IEEE Access*, 12:197766–197782, 2024

II T. Treier. Beyond the happy path: A safety-critical audit methodology for Estonia's i-voting processing application. *IEEE Access*, 13:203429–203443, 2025

III T. Treier. Re-voting under surveillance: National eID transaction logs as a threat to coercion resistance in Estonian internet voting. In D. Duenas-Cid, P. Roenne, M. Volkamer, M. Blom, P. Gaudry, I. Borucki, L. Loeber, and A. Debant, editors, *Electronic Voting*, pages 191–207, Cham, 2025. Springer Nature Switzerland

Author's Contributions to the Publications

I For Publication I, I was the main author. I initiated the research, formulated the integrity problem in the processing stage, and derived the artefact-based verification logic (the integrity invariant). My co-author implemented the prototype auditing script and supported the setup of the test environment (as part of an M.Sc. project under my supervision). I validated the approach and results, prepared the figures, and wrote the manuscript.

II For Publication II, I was the sole author. I developed the safety-critical audit methodology, mapped functional requirements to fault scenarios, and constructed the synthetic fault-seeded test corpus. I defined explicit detection oracles, conducted the experiments to evaluate audit-tool sensitivity, and wrote the manuscript.

III For Publication III, I was the sole author. I identified the architectural tension between the revoting mechanism and the national eID logging infrastructure. I developed the threat model, analyzed the metadata-based side-channel leakage, proposed mitigation strategies, and wrote the manuscript.

Abbreviations

DO-178C	Software Considerations in Airborne Systems and Equipment Certification
DO-333	Formal Methods Supplement to DO-178C and DO-278A
E2E	End-to-End (as in end-to-end verifiability)
eID	electronic identity
i-BB	Internet ballot box
i-voting	Internet voting
IV&V	Independent Verification and Validation
IVXV	Estonian Internet Voting Framework
NASA	National Aeronautics and Space Administration

Terms

anonymous ballot box	The processed output collection containing only e-ballots, intended to be unlinkable to voters and suitable for tallying.
audit artefacts	Election artefacts made available for auditing, such as signed containers, ballot-box snapshots, lists, and logs released according to the election procedure.
audit sensitivity	A measurable property of an audit procedure or toolchain indicating which modelled fault or attack classes it can detect under a stated evidence surface and oracle set.
audit toolchain	The set of tools, scripts, procedures, and checks used to evaluate election artefacts during an audit.
ballot	A technical artefact that represents a vote within the system at the data level.
coercion resistance	A security property requiring that an adversary cannot reliably determine whether a voter complied with coercion, even if the voter is forced to cooperate by revealing available evidence.
electronic ballot (e-ballot)	An anonymised encrypted ballot payload obtained during processing by separating the payload from voter-linked data. E-ballots form the anonymous input to tallying and mixing and are intended to be unlinkable to voters.
electronic vote (e-vote)	A voter-associated voting artefact collected during the voting phase in IVXV and stored as a signed container. An e-vote contains an encrypted ballot payload together with voter-linked data that enables eligibility checks and later invalidation under revoting and paper-ballot override rules.
end-to-end (E2E) verifiability	An assurance concept in which voters and public observers can check, through system-provided evidence, that eligible votes are correctly captured, recorded, and reflected in the tally while ballot secrecy is preserved.
evidence surface	The subset of audit artefacts, together with the defined normalisation and linking rules over them, that constrains what an independent auditor can check without trusting internal execution.
explicit oracle	An audit oracle stated in a form that is reproducible and machine-checkable, rather than left to discretionary or informal judgement.
fault scenario	A modelled deviation class representing a specific fault, attack, or anomalous behaviour that an audit procedure may or may not detect.
fault-seeded corpus	A synthetic collection of test datasets in which deviations are deliberately introduced so that audit sensitivity can be evaluated against known fault classes.
integrity invariant	

(accounting invariant)	A formally stated balance relation over audit artefacts requiring that every input ballot is either preserved in the anonymous output or explicitly accounted for through an allowed invalidation path.
Internet ballot box (i-BB)	The collection of accepted e-votes from the voting phase, stored as signed containers and used as the primary input to the processing stage.
invalidated e-vote	An e-vote excluded from the final tally and therefore diverted from the anonymous output, with the reason recorded in explicit invalidation artefacts or validation outputs.
invalidation (invalidate)	A protocol-prescribed act of excluding an e-vote from contributing to the final tally, including revoting-based supersession and paper-ballot override. Technical invalidity is treated separately in the accounting model.
key application	An IVXV software component used for election key-management operations, including generation and handling of cryptographic key material in accordance with the election procedure.
metadata side channel	A leakage channel in which information is inferred not from ballot content itself but from auxiliary metadata, such as counts, timing, or event patterns.
mutation score	A metric expressing how many fault-seeded variants (mutants) are detected by a tool or audit procedure relative to the number exercised.
oracle (audit oracle)	A deterministic predicate over the evidence surface that defines what constitutes a detectable deviation in a given audit test case.
processing application	The offline software component that executes the processing-stage pipeline and emits signed intermediate artefacts and logs used as audit evidence.
processing stage	The offline pipeline that transforms the i-BB into an anonymous ballot box by (i) applying protocol-prescribed invalidation rules and (ii) anonymising ballots by removing voter-linked data prior to counting.
receipt-freeness	A weaker privacy property requiring that a voter cannot produce transferable evidence of how they voted that would convince a coercer or vote buyer.
revoting	Casting multiple e-votes such that only the last eligible e-vote is counted. A later paper vote overrides the e-vote(s).
revoting secrecy	The assumption that whether and when a voter revoted is not reliably inferable by an adversary from available evidence.
software independence	The property that outcome-changing software faults or malicious software changes cannot alter the election outcome without producing detectable evidence that enables the change to be discovered through auditing.

system composition	The interaction of the voting system with surrounding technical and institutional infrastructures whose combined behaviour may affect security properties even when each component is analysed separately.
technical invalidity	Invalidity due to cryptographic or structural failures, such as malformed containers, invalid signatures, invalid encryption, or inconsistent metadata, discovered during validation and handled separately from revoting and paper override.
vote	The voter's intended choice at the semantic level, i.e., what the voter means to express.

Symbols

B_1	Signed input e-vote collection (i-BB contents) at the start of processing.
E	E-votes removed due to technical invalidity discovered during validation.
B_2	Anonymous output e-ballot collection produced for tallying and mixing.
L_1	E-votes invalidated due to revoting (e.g., squash output or revoting log).
L_2	E-votes invalidated due to paper-ballot override (e.g., revoke output or override log).
R_i	Functional requirement i as defined in Publication II.
F_i	Fault scenario i as defined in Publication II.
O_i	Audit oracle i as defined in Publication II.
$\setminus$	Set difference.
$\cup$	Set union.

1 Introduction

Internet voting (i-voting) poses a difficult assurance problem. Election outcomes must be independently checkable, yet ballot secrecy must remain protected in an uncontrolled environment [1, 2]. In Estonia, this problem is operational rather than hypothetical, because i-voting has functioned as a long-standing nationwide voting channel across multiple election cycles [5, 35]. Estonia therefore provides a consequential case study for analysing how trust in i-voting can be justified from evidence rather than assumed from procedure alone.

This dissertation examines the server-side processing stage of Estonia's i-voting system and the auditing practices used to evaluate it. Its central question is not whether the system should be trusted in a general sense. Rather, it asks what kinds of independently checkable evidence are needed to justify trust in the processing stage, in the audit tools used to evaluate it, and in the broader assumptions that make this processing logic necessary. Throughout the dissertation, trust is used in this narrower assurance sense: justified confidence in specific technical claims based on independently checkable evidence. It is not used as a claim about the broader sociological formation of public or institutional trust, although such trust may be affected by the availability and credibility of evidence. This chapter introduces the assurance problem addressed in the dissertation, motivates the work through the research path from which it emerged, defines the research objectives and methodological approach, and provides a compact map from research questions to publications.

1.1 Problem Statement and Motivation

The Estonian i-voting system is embedded in a broader national e-government ecosystem and has been used in multiple parliamentary and local election cycles [40, 5]. Despite this maturity, public and political discussion has repeatedly questioned whether i-voting is sufficiently trustworthy to remain a nationwide voting channel [6, 7]. This occurs even though the share of voters using i-voting has remained high and has, in some elections, exceeded half of all participating voters [35]. The resulting tension reflects a core assurance problem. How, and at what level of detail, can the correctness of the system be demonstrated in a way that is independently convincing?

To address this problem, the dissertation makes three linked contributions. First, it defines an artefact-based integrity relation for processing-stage auditing. Second, it develops a safety-critical methodology for validating audit tooling. Third, it examines whether the secrecy assumptions that motivate revoting-driven processing complexity remain valid when the voting system is composed with the surrounding eID ecosystem.

The dissertation does not propose a new voting protocol. Rather, its contribution lies in reframing the auditability of an operational i-voting system as an evidence-based testing problem over artefacts, audit tools, and surrounding system assumptions. Although the dissertation is grounded in the Estonian case, its contribution is methodological: it develops assurance concepts and audit strategies that are relevant whenever outcome-critical election software must be checked from externally observable evidence.

Heiberg et al. [14] initially presented IVXV as satisfying end-to-end verifiability requirements. Later work in the same research line qualified this view by arguing that the current Estonian i-voting scheme should not be read as satisfying the strongest interpretations of end-to-end verifiability in isolation from secrecy and coercion-resistance constraints [13]. The perspective adopted in this dissertation is complementary and explicitly engineering-driven. This dissertation focuses on whether assurance claims are operationally instantiated

through independently checkable artefacts, validated audit procedures, and surrounding system assumptions. If the software logic is defined and the input artefacts are fixed, then the outcome-relevant transformation should be reproducible and verifiable by an independent party. Trust should therefore not rest solely on the assumption that procedures operate correctly. It should also rest on the ability to check correctness from observable artefacts and evidence, in line with the broader idea of software independence, namely that outcome-changing faults must be detectable from evidence that can be checked independently of the system under test [27].

The dissertation originated in a laboratory deployment of key components of Estonia's central election system based on publicly available source code. This work exposed a critical gap in how the processing stage could be audited independently. The processing stage is the phase in which collected voter-linked voting artefacts are filtered according to revoting and paper-voting rules and are then anonymised before counting [32, 34]. From an assurance perspective, this stage forms an evidential bottleneck. It is the last stage at which an auditor can still relate outcome-relevant transformations to voter-linked inputs before the ballots are intentionally detached from voter identity. After anonymisation, correctness can still be assessed at the aggregate level, but the traceable relation between individual input artefacts and counted ballots is intentionally lost.

In practice, this stage offered only limited possibilities for independent artefact-based checking by external observers and auditors. To the best of the author's knowledge, there was no general, formally motivated control that would link the input set of voter-linked artefacts to the anonymised output while accounting only for protocol-prescribed invalidations. Under this gap, a faulty implementation or a malicious insider could in principle substitute ballots during processing in a way that later checks might not reliably detect, as demonstrated in Publication I [39].

The first phase of the research therefore focused on defining a checkable integrity relation for the processing stage. The resulting control is expressed as a data-driven input–output invariant over processing-stage artefacts and is designed to be evaluated without relying on the internal execution logic of the processing application. Its purpose is to show that every ballot entering processing is either preserved into the anonymised output or explicitly invalidated through an allowed protocol path.

When this logic was taken towards practical use, however, an important limitation became visible. As demonstrated in Publication II [37], one deliberately crafted manipulation passed the log-based check without raising an alert. This indicated that an integrity check alone is not sufficient for a credible audit argument. Auditors must also be able to justify that the audit tool is sensitive to the fault and attack conditions that matter in practice. This motivated the second phase of the research and the development of a systematic audit methodology inspired by safety-critical assurance practice, in which the audit tools themselves are tested using deliberate fault seeding, explicit oracles, and structured coverage analysis, as developed in Publication II.

A further insight emerged from this work. A substantial part of the processing stage's complexity stems from supporting revoting, which in Estonia is a primary mechanism for protecting voting freedom against coercion in an uncontrolled environment [15, 16]. That complexity is therefore not incidental. It exists because the system attempts to preserve a secrecy-based coercion-mitigation mechanism while still allowing later auditing of outcome-relevant transformations. However, system-level analysis suggests that revoting secrecy may be weakened by the national electronic identity (eID) ecosystem on which the voting system depends. In particular, eID transaction logs can form a metadata side channel that enables revoting events to be detected. If the number and timing of vote-related

actions remain observable through durable eID log records, then the secrecy assumptions that justify revoting-driven processing complexity must be re-evaluated. Publication III [38] addresses this problem by showing how component interaction can weaken coercion resistance even when the individual components serve legitimate purposes.

Taken together, the dissertation develops an assurance framework with three linked layers. First, processing integrity is constrained through explicitly checkable artefact relations. Second, audit tooling is subjected to evidence-based validation so that its detection capability becomes measurable rather than assumed. Third, the rationale for processing-stage complexity is interpreted compositionally by examining whether revoting secrecy still holds under interaction with the surrounding eID ecosystem.

These three contributions support a two-part central claim. On the one hand, processing-stage integrity and audit sensitivity can be supported rigorously through artefact-based modelling, explicit oracles, and fault-based evaluation. On the other hand, coercion-resistance claims about revoting cannot be interpreted in isolation from external metadata flows.

This line of research has already been reflected in practical developments. The integrity-check logic developed in Publication I was incorporated into the audit tooling released for the 2024 European Parliament elections and published in the State Electoral Office of Estonia's public repository [33]. Furthermore, following Publication II, the State Electoral Office of Estonia has publicly stated that it is developing, for the 2027 parliamentary elections, a software solution that enables the auditor to independently repeat the processing-stage checks performed by the election authority. The same public statement also describes validating the audit tool using fault descriptions and test datasets [31].

1.2 Research Objectives and Approach

The overall objective of this dissertation is to develop and validate a methodology that treats auditing of Estonia's i-voting processing stage as an evidence-based, safety-critical testing problem rather than a procedure grounded primarily in trust. The dissertation operationalises this objective by stating checkable claims over the election artefacts made available for auditing and by validating the sensitivity of audit tooling against explicitly modelled fault and attack conditions.

The first sub-objective is to formulate a checkable integrity relation between input and output artefacts of the processing stage. The processing stage is modelled so that an auditor can verify integrity at the level of the available input and output artefacts, without trusting the internal logic of the processing application. Within the model assumptions, no ballot may disappear, be substituted, or be added except through protocol-prescribed invalidations such as revoting or paper-voting override. This sub-objective is addressed in Publication I.

The second sub-objective is to develop a safety-critical audit methodology for evaluating the sensitivity and coverage of auditing tools. Based on analysis of documentation and source code, functional requirements and potential fault conditions are modelled. From this model, synthetic test datasets are generated with deliberately seeded faults, and explicit detection oracles are defined for automated evaluation. This makes it possible to quantify which classes of attacks and faults a given audit process can detect and which risks remain outside its demonstrated detection capability. This sub-objective is addressed in Publication II.

The third sub-objective is to analyse, at the system level, the interaction between revoting and eID logs. The dissertation examines whether the secrecy assumptions that motivate revoting-driven processing complexity remain valid when the voting system is

composed with the surrounding eID ecosystem. The objective is to identify the leakage channel and to assess how it constrains the interpretation of revoting-driven transparency and assurance measures. This sub-objective is addressed in Publication III.

To address these objectives, the dissertation is structured around three research questions:

RQ1. How can processing-stage integrity be verified from the input and output audit artefacts while accounting only for protocol-prescribed invalidations, without trusting the internal logic of the processing application?

RQ2. How can safety-critical software testing be applied to measure the sensitivity and coverage of audit tooling against explicitly modelled fault and attack scenarios?

RQ3. How does revoting secrecy change under composition with eID transaction logs that may be disclosed after elections, and what are the implications for coercion resistance and for the rationale of processing-stage assurance?

Methodologically, the dissertation combines mathematical modelling, safety-critical testing techniques, and system-level composition analysis into a single framework. For RQ1, set-theoretic modelling is combined with scenario-based validation to define an artefact-based integrity invariant. For RQ2, auditing is treated as a test target using fault seeding, explicit oracles, and measurable sensitivity outcomes. For RQ3, the methodology applies a leakage model over transaction-log metadata and evaluates mitigation strategies under deployability and governance constraints. Together, the dissertation moves from artefact-level integrity to assurance of audit tooling and finally to the system-level justification of the processing complexity those controls are meant to supervise.

The scope of this dissertation is limited to the server-side processing stage and its auditing process in Estonia's i-voting system (IVXV). The dissertation does not attempt a detailed treatment of client-side security problems, user-interface risks, network- and infrastructure-level attacks, or the broader political and sociological debate around i-voting. These issues are considered only insofar as they are necessary for interpreting processing-stage artefacts, revoting, and the surrounding assurance assumptions. The dissertation also does not aim to provide an exhaustive comparison of international i-voting schemes. Instead, Estonia is used as a detailed case study through which broader methodological conclusions are developed in later chapters.

The results of this dissertation are interpreted under three boundary assumptions. First, auditors can obtain and examine the relevant signed containers, ballot box snapshots, and logs produced by the election process. Second, the execution environment of the processing application is not assumed trustworthy. Third, the analysis addresses specific threat classes rather than claiming universal coverage. For processing-stage integrity, the main concern is faulty software or an insider capable of manipulating artefacts or their transformations. For coercion resistance, the main concern is a coercer who can later compel a voter to reveal eID transaction history and infer revoting behaviour from metadata such as counts and timestamps.

Table 1 provides a compact map that links the research questions to the main technical idea, evidence surface, and the corresponding publication. The map is intended to show that the three publications do not form an arbitrary collection. Rather, they contribute to a single assurance argument that progresses from checkable processing integrity, to demonstrated audit sensitivity, and finally to a compositional reassessment of the assumptions that justify revoting-driven complexity.

Table 1: Thesis map from research questions to approach and publications.

RQ	Core idea and evidence surface	Publication
RQ1	Artefact-based accounting invariant that links signed processing inputs to anonymised outputs while making allowed invalidation paths explicit and checkable from the audit artefacts.	Pub. I
RQ2	Safety-critical audit validation of tooling using requirements–fault modelling, a synthetic fault-seeded corpus, explicit detection oracles, and measurable sensitivity outcomes.	Pub. II
RQ3	System-level composition analysis of coercion resistance under revoting, focusing on metadata leakage via eID transaction logs, a leakage model for detecting revoting events, and mitigation strategies for reducing or encapsulating the side channel.	Pub. III

1.3 Structure of the Dissertation

Following this introductory chapter, the remainder of the dissertation proceeds as follows:

- Chapter 2 (Background) establishes the assurance concepts and technical context required to interpret the results, including E2E verifiability, software independence, coercion resistance, the IVXV processing stage, and the role of revoting and eID transaction logs.
- Chapter 3 (Results) summarises the results reported in the three publications, including the integrity invariant and its application, the safety-critical audit methodology and its evaluation logic, and the system-level analysis of revoting and eID logs.
- Chapter 4 (Discussion) synthesises the results into a coherent whole, connects the contributions of the three underlying publications through an overarching narrative, and discusses implications for auditing practice, system governance, generalisability, and limitations.
- Chapter 5 (Conclusion) summarises the dissertation's scientific contributions, provides concise answers to the research questions, and outlines the most important practical recommendations and directions for further research.

2 Background

This chapter provides the theoretical and technical framework needed to interpret the dissertation results. It introduces the assurance concepts that motivate evidence-based auditing in i-voting, with emphasis on end-to-end (E2E) verifiability, software independence, and coercion resistance. It then describes the Estonian IVXV system, focusing on the processing stage as an evidential bottleneck between voter-linked input artefacts and anonymous output artefacts [32, 34]. Next, it outlines safety-critical testing and auditing principles that motivate the "testing for trust" approach used in this work. Finally, it explains revoting as Estonia's primary practical mechanism for mitigating undue influence in an uncontrolled environment and shows why the national eID ecosystem and its transaction logs create a systemic tension with ballot secrecy.

2.1 Assurance concepts and evidence in i-voting

A central challenge of i-voting is to reconcile two demanding requirements. Election outcomes must be independently checkable, yet ballot secrecy must remain protected. End-to-end (E2E) verifiability is used here as an umbrella term for mechanisms intended to support independent checking of election outcomes without compromising ballot secrecy. At a high level, such mechanisms seek to provide voter-level assurance that a vote was correctly captured and recorded, and public assurance that all eligible votes are correctly reflected in the final tally without revealing how any individual voted [1, 4, 2]. This dissertation does not rely on one strongest formal definition of E2E verifiability. Instead, it treats E2E verifiability as a family of assurance goals whose concrete interpretation depends on the balance between public checkability, ballot secrecy, and coercion resistance.

As the wider E2E-verifiability literature has matured, the interpretation of verifiability in the Estonian i-voting literature has also been qualified. Earlier work on IVXV presented the scheme as satisfying E2E-verifiability requirements, whereas later work in the same research line argued that the current Estonian i-voting scheme should be understood as balancing verifiability against secrecy and coercion-resistance constraints rather than as satisfying the strongest interpretations of E2E verifiability outright [14, 13]. This dissertation does not simply reject the earlier line of work. Rather, it distinguishes between verifiability as a scheme-level property and assurance as an operational, evidence-based argument grounded in released artefacts, validated audit procedures, and explicit evidence surfaces. From this perspective, a system may incorporate E2E-verifiability mechanisms and still leave open important practical questions about whether critical transformations are independently checkable in deployed audits, whether audit-tool sensitivity has been demonstrated, and how narrow the verifier community remains in practice.

More broadly, the literature on i-voting shows that verifiability is not only a protocol property. Real assurance depends on how evidence is produced, disclosed, interpreted, and audited. Weak procedures can undermine otherwise sound cryptographic mechanisms [4, 3]. A detailed case study shows that verification arguments can fail in practice when the assumptions underlying the verification do not hold in the deployed system, when the checked facts are insufficient to imply the claimed property, or when verification procedures appear to pass despite manipulated votes [12].

A closely related concept is software independence. Rivest defines software independence as the property that no software error or malicious software change can alter the election outcome without being detected [27]. If the outcome changes, the change must leave behind detectable evidence [27]. Importantly, software independence does not forbid the use of software in verification. Rather, it shifts the trust basis toward independently

checkable artefacts, invariants, and auditable transformations instead of assuming that a particular application executed correctly [27, 4].

Beyond correctness, i-voting systems must also address coercion and undue influence. Coercion resistance is stronger than receipt-freeness, because even if a voter is forced to cooperate, the adversary should not be able to determine whether the voter complied [15]. In the literature, revoting is proposed as a practical coercion-mitigation mechanism for i-voting. A coerced vote can later be overridden by voting again in a private setting [15, 16]. In Estonia, this mechanism is tightly coupled to server-side processing rules and to the limits of what evidence can be safely disclosed without weakening secrecy, because the system must invalidate earlier electronic votes (e-votes) and may also invalidate an e-vote in favour of a paper ballot [16, 32, 34].

This dissertation adopts an evidence-based interpretation of these concepts. It treats assurance as a claim supported by deterministic checks over the artefacts released for auditing rather than by trusted execution alone [27]. In that sense, E2E verifiability, software independence, and coercion resistance are treated here as practical constraints on what evidence must exist, what evidence can be disclosed, and what deviations an audit should be able to detect.

2.2 IVXV and the processing stage as an evidential bottleneck

The current Estonian i-voting implementation is based on the IVXV platform, which integrates the practical election process with verifiability-related mechanisms and supporting procedures [14, 32, 34]. The system lifecycle is typically described via four phases: setup, voting, processing, and counting [32, 34]. Because this dissertation focuses on auditability between observable artefacts and the semantic rules applied to them, the processing stage is the key phase.

In processing, the Internet ballot box (i-BB) containing encrypted and signed e-votes is transformed into an anonymous and countable output by applying invalidation rules necessitated by revoting and paper voting [32, 34]. Processing therefore combines two assurance-critical functions. First, it enforces eligibility and invalidation rules. Second, it produces anonymity by separating voter-linked data from ballot payloads [32, 34]. From an audit perspective, the processing stage is an evidential bottleneck because it is the last stage at which outcome-relevant transformations can still be related to voter-linked inputs before the ballots become unlinkable.

Figure 1 visualises the processing steps and the flow of artefacts from voter-linked inputs to anonymous outputs [32].

Within IVXV, processing is performed in an offline environment and consists of the following main steps [32, 34]:

- **Check:** Verify the integrity of the i-ballot box by checking vote signatures, registration-service confirmations, and timestamps, and by cross-checking the collected votes against the registration data.
- **Squash:** Identify, for each voter, the last valid e-vote and invalidate earlier e-votes cast by the same voter.
- **Revoke:** Apply signed revocation and restoration lists, including the invalidation of e-votes where a later paper vote takes precedence under parallel voting rules.
- **Anonymise:** Group the remaining e-votes by district and remove links between voters and ballot payloads while preserving the district association needed for counting.

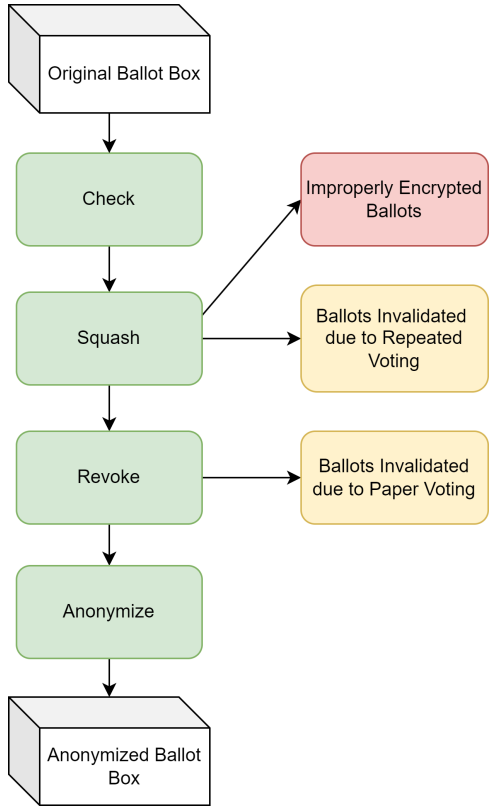

Figure 1: The steps and dataflow of the processing stage (Check–Squash–Revoke–Anonymise), including classes of invalidated votes, reproduced from Publication I and originally adapted from [30].

- **Mix:** Cryptographically permute and re-randomise the anonymous ballots to break input–output order links and produce a mixing proof for audit.

Because some votes are intentionally invalidated, a naive expectation that the input and output ballot boxes should simply "match" will always fail. Auditability therefore depends on defining an artefact-level integrity relation that makes legitimate invalidations explicit and independently checkable. An auditor needs a normalisation and accounting model that makes each invalidation class explicit and checkable, so that deviations become detectable rather than explainable away as an expected mismatch. Processing is executed by a dedicated offline software component, here referred to as the processing application. It orchestrates the pipeline and emits signed intermediate artefacts and logs, which together define the primary evidence surface on which a data-driven audit can operate [32, 34].

Technical invalidity is treated separately, for example through removal during **Check**, whereas revoting and paper override are handled as explicit invalidation classes in **Squash** and **Revoke** [32, 34]. This distinction matters because it determines what kinds of ballot disappearance are legitimate and what kinds would constitute an integrity violation. This dissertation focuses on processing-stage artefacts because they constrain what an independent auditor can check without relying on the processing application's internal execution.

2.3 Testing for trust as safety-critical audit validation

Auditing an election system differs from conventional information-security assessment because the potential failure is not merely technical. A failure can damage institutional legitimacy and undermine trust in the electoral process [18]. Although i-voting is not life-critical in the narrow technical sense, its societal criticality is comparable, because a silent failure can delegitimise elected authority, destabilise institutions, and erode public trust for years. This makes mission- and safety-critical assurance principles relevant as an engineering analogy, especially where assurance should rest on layered and independently checkable evidence rather than on procedural confidence alone [10, 4]. The literature on end-to-end verifiable i-voting has likewise argued that such systems for public elections should be designed, constructed, verified, certified, operated, and supported according to the most rigorous engineering requirements of mission- and safety-critical systems [4]. This dissertation adopts that perspective by treating the audit toolchain as a test target and by validating audit sensitivity against explicitly modelled fault and attack conditions. More broadly, safety- and mission-critical engineering emphasises systematic evidence, explicit assumptions, and disciplined verification activities in order to reduce the probability that latent faults survive into operation [10].

In such domains, assurance practice commonly includes requirements-based verification, independent review or assessment functions, and documented coverage objectives [41, 20, 25]. The question is not only whether the system behaves correctly under nominal conditions. It is also whether defined fault classes and off-nominal deviations are detectable [41, 17]. NASA's Independent Verification and Validation (IV&V) guidance emphasises independent evidence and the need to exercise both nominal and off-nominal scenarios [20]. Similarly, certification-oriented interpretations of DO-178C stress linking tests to requirements and accounting for coverage as part of certification-grade verification [25]. DO-333 further discusses the role of formal methods as an alternative route to satisfying comparable assurance objectives and evidence obligations [19].

These ideas are particularly relevant in i-voting because the audit is typically data-centric. Audit inputs are logs, signed containers, and ballot box snapshots rather than live execution in a fully controlled laboratory environment [32, 34, 4, 12]. If the execution environment cannot be fully trusted, the audit must define explicit oracles and invariants over the artefacts that are actually available and that can be reproduced independently [20, 27]. In this dissertation, an explicit oracle is a deterministic predicate over the audit artefacts that defines what constitutes a detectable deviation.

One practical way to evaluate the evidential power of an audit is fault seeding and mutation testing [22]. Instead of waiting for random faults, one constructs anomalous inputs or artefact mutants that model known fault classes and then checks whether the audit procedure detects them [22]. This makes it possible to report audit sensitivity in measurable terms and to discuss coverage not only with respect to requirements but also with respect to modelled fault classes [22]. From this perspective, claims of verifiability do not depend only on protocol design. They also depend on the measurable sensitivity of the auditing process to the fault and attack classes that matter in practice [4, 3, 12].

2.4 Revoting secrecy under composition with the eID ecosystem

A distinctive feature of Estonian i-voting is revoting. A voter may cast multiple e-votes, of which only the last is counted, and a paper vote may override an e-vote [36, 26, 32, 34]. In Estonia, revoting is motivated as a practical mitigation against undue influence in an uncontrolled environment and therefore interacts directly with both secrecy goals and

evidential design choices [36, 16, 15].

A crucial assumption behind revoting as a coercion-mitigation mechanism is that an attacker has no reliable way to observe or prove whether and when a voter revoted. Studies emphasise that metadata leakage channels are critical here. If revoting becomes externally observable at scale, the anti-coercion value of revoting degrades [16].

In Estonia, i-voting relies on a national eID ecosystem in which authentication and digital signing are delegated to qualified trust-service providers [8, 9, 29]. This creates a systemic composition problem. From the election-system perspective, anonymity and separability become essential after a certain point in the process [32, 34]. From the trust-service perspective, the ecosystem is designed to support non-repudiation, accountability, and auditability, including record-keeping and event-logging obligations [8, 9, 29].

Viewed as a composition of systems, the risk does not necessarily arise from a weakness in a single component. Instead, combining components with different security objectives can yield auxiliary information that was not intended in either design [28, 11]. This dynamic is demonstrated for the Estonian revoting mechanism and eID transaction logs in Publication III. In the Estonian case, eID transaction logs can reveal the number and timing of vote-related signing events even though they reveal no ballot content. That information can function as a metadata-based signal showing whether and when a revote occurred.

This tension matters for audit interpretation as well. A substantial part of processing-stage complexity exists in order to support revoting and paper-based overrides while preserving both verifiability and secrecy constraints [16, 32, 34]. If revoting secrecy is weakened by persistent metadata in the surrounding eID ecosystem, the assumption that justifies revoting as a practical anti-coercion mechanism, and with it part of the processing-stage complexity, must be reassessed under composition rather than only within the election protocol itself.

This dissertation therefore analyses revoting secrecy under system composition. Coercion resistance depends on information flows across institutional and technical boundaries that the election protocol does not fully control.

3 Results

This chapter summarises the dissertation's contributions as reported in Publication I, Publication II, and Publication III. The presentation follows a widening scope: from a concrete artefact-level integrity check for the processing stage, to a methodology for evaluating audit sensitivity via fault seeding and explicit test oracles, and finally to a system-level analysis of how revoting secrecy can be undermined by the surrounding eID logging ecosystem. The goal of the chapter is descriptive rather than argumentative. Interpretation, synthesis, and broader implications are developed in the subsequent discussion chapter.

3.1 A checkable integrity invariant for the processing stage (Publication I)

This section reports the first contribution, which addresses a specific auditability gap in the IVXV processing stage. The emphasis is on an artefact-level integrity relation that can be checked independently using the artefacts released for auditing. The section first motivates the gap, then states the invariant and normalisation approach, and finally summarises detection capability and documented limitations.

3.1.1 Auditability gap and contribution

The first result addresses a concrete auditability gap in the IVXV processing stage. Processing must both (i) apply invalidation rules related to revoting and paper-ballot override and (ii) break the link between voter identities and encrypted ballots prior to anonymous counting. Consequently, an external auditor cannot rely on signatures or successful pipeline termination alone. What is required is an artefact-level integrity relation that accounts for every ballot from the signed input to the anonymised output while making all allowed invalidation paths explicit and independently checkable on the basis of the audit artefacts.

3.1.2 Artefact model and the invariant

Publication I models the relevant artefacts as checksum-normalised sets:

- B_1: the signed input ballot box (i-BB contents);
- E: ballots removed due to technical invalidity (e.g., invalid encryption);
- B_2: the anonymised output ballot box;
- L_1: ballots invalidated by revoting ("squash" log);
- L_2: ballots invalidated due to paper-ballot override ("revoke" log).

The proposed integrity invariant is:

$$B_1 \setminus E \;=\; B_2 \cup L_1 \cup L_2. \tag{1}$$

Intuitively, after removing technically invalid ballots (E), every remaining input ballot must either appear in the anonymised output (B_2) or appear explicitly in one of the invalidation artefacts (L_1 or L_2). This also clarifies terminology. Revoting- and paper-related invalidation is accounted for via L_1 and L_2, while technical invalidity is treated separately via E. The invariant is evaluated together with the side condition that $E \subseteq B_1$ in the normalised identifier space. An identifier appearing in E without a counterpart in B_1 would therefore be a separate artefact-consistency anomaly, not a valid explanation for removing an input ballot.

To make the comparison independent of internal data structures, Publication I specifies normalisation steps that map heterogeneous artefacts onto a shared identifier space at

the cryptogram checksum level before set comparison. Figure 2 illustrates these transformations and the resulting artefact links.

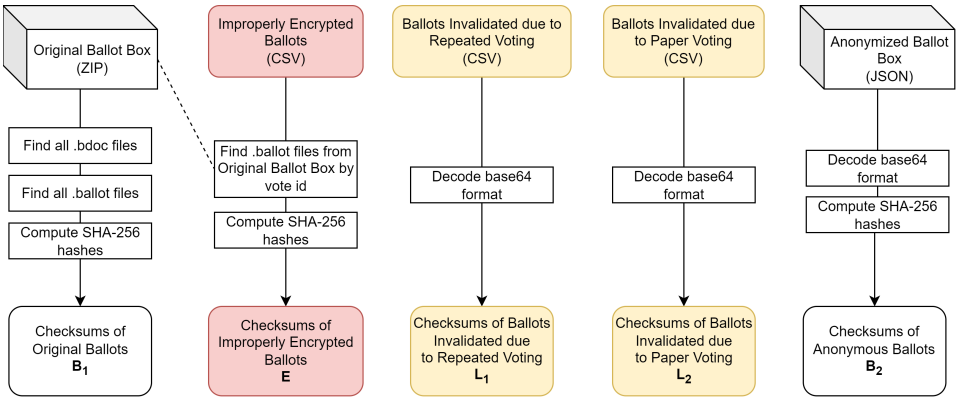

Figure 2: Transforming ballots for comparison (reproduced from Publication I).

3.1.3 Detection capability, boundary conditions, and practical uptake

Publication I implements Equation (1) as an independent audit script and evaluates it on nominal and manipulated cases. The invariant detects integrity violations in which ballots are added, removed, or replaced in the anonymised output or invalidation artefacts without a corresponding explanation in E, L_1, or L_2.

The publication also makes two limitations explicit. First, because the invariant is an accounting relation over a union, it does not by itself guarantee that ballots are correctly categorised. For example, moving a ballot between B_2 and L_2 may preserve the union while still being semantically wrong. Second, because the comparison is set-based, a uniqueness assumption must be handled explicitly. Accordingly, the publication recommends complementing the set check with a multiplicity or duplicate control where duplicate insertion is a plausible manipulation.

Operationally, the publication reports that the findings were communicated to the election authority and that the enhanced audit check was incorporated into publicly released IVXV audit tooling. This practical uptake suggests that the artefact-based framing is not only theoretically motivated but also operationally tractable.

3.2 Safety-critical auditability of the processing application (Publication II)

This section reports the second contribution, which shifts focus from a single integrity predicate to the audit toolchain as an object of evaluation. The key idea is that an audit procedure can be assessed for sensitivity against a stated fault model using fault seeding and explicit oracles. The section introduces the modelling elements, namely requirements, fault scenarios, and oracles, and then reports Phase A coverage and sensitivity results for an exemplar corpus.

3.2.1 From “auditing the system” to “testing the audit”

The second result generalises the first by asking how an auditor can justify that the audit toolkit itself is sensitive to relevant deviation classes. Publication II adopts a safety-critical testing framing and treats audit mechanisms as test targets. The methodology combines

requirements–fault mapping, synthetic corpus construction with fault seeding (mutation), and explicit, machine-checkable test oracles.

3.2.2 Requirements, fault scenarios, and explicit oracles

Based on documentation and code analysis, Publication II identifies 19 functional requirements (R_1–R_{19}) and 37 fault scenarios (F_1–F_{37}). A central contribution is the use of explicit oracles, that is, deterministic predicates over the available audit artefacts, so that detection claims do not rely on subjective judgement. Publication II defines six primary oracles O_1–O_6 spanning set consistency, last-vote validity, paper-override consistency, district-mapping preservation, signature or packaging validity, and cross-artefact consistency.

3.2.3 Phase A coverage and sensitivity snapshot

Publication II reports Phase A measurements for the exemplar corpus, covering both mapping coverage and exercised tool sensitivity. However, the presence of one or more mapped fault classes should not be interpreted as proof of exhaustive requirement coverage; it indicates only that at least one aspect of the requirement is exercised within the current corpus scope. The reported Phase A figures are summarised in Table 2, with percentages shown for readability.

Table 2: Phase A coverage and sensitivity snapshot (from Publication II).

Metric	Definition (as used in Publication II)	Result
R-coverage	Requirements with ≥ 1 mapped fault class in the current requirement–fault mapping within the Phase A corpus scope	$12/19$ (63%)
Mutation score (tool under test)	Mutant variants detected ("killed") by the publicly released log-based integrity-check tool over the exemplar corpus	$5/17$ (29%)
Within-scope sensitivity (O_1)	Detection of exercised set-consistency anomalies within the tool's stated scope (O_1); one miss is reported ($F_{26\text{-}2}$)	$5/6$ (83%)
Out-of-scope sensitivity (O_2–O_6)	Detection on exercised variants targeting O_2–O_6 (last-vote validity, paper override, district mapping, signature/packaging, cross-artefact consistency)	0

These figures characterise the exemplar corpus and the tested tool, not the operational IVXV system or official audit practice. In particular, the zero score outside O_1 is an expected result given the tool's evidence surface and stated scope. A log-based set-consistency checker is not designed to detect deviations that require different evidential hooks, such as district mapping or paper-override semantics.

3.2.4 Interpreting the instructive miss and the mapping presentation

Publication II highlights the within-scope miss $F_{26\text{-}2}$ as especially instructive. The scenario extends unique ballot addition by also editing the same logs and aggregates on which the tool relies, so that the released totals remain consistent with the altered output state. This probes whether an audit binds aggregate evidence to per-ballot reality rather than accepting aggregate alignment alone. The case therefore motivates the need for additional independent oracles beyond a single log-based set check.

Because the full $R \times F$ mapping is large, the thesis presents in the main text only a

compact orientation view aligned with the oracle structure. The complete fault catalogue, the full functional requirements list, and the dissertation-level requirement–fault mapping used for coverage planning are provided in Appendix 4. Table 3 provides a concise oracle-aligned summary for the main text.

Table 3: Representative requirement–fault mapping patterns organised by oracles (summary view based on Publication II).

Oracle	What the oracle checks (artefact-level)	Representative fault classes (examples)
O_1 Set consistency	No unexpected additions, removals, or duplicates in defined sets, such as i-BB entries versus registration lists or ballot-accounting relations.	Input/list mismatches and duplication anomalies; output-integrity deviations such as adding, removing, or duplicating ballots (F_{26}–F_{30}).
O_2 Last-vote validity	For non-paper voters, exactly one ballot remains valid and it is the latest; paper-voter cases are handled separately by O_3.	Ordering and validity manipulations for multi-voters (F_{33}–F_{36}).
O_3 Paper override consistency	For paper voters, all e-votes are invalidated in processed outputs.	Failure to invalidate despite a paper vote (F_{37}).
O_4 District mapping preservation	Anonymised outputs preserve correct electoral-district attributes derived from input lists.	Incorrect district assignment and other district-related output-integrity issues, including F_{25} and F_{32}.
O_5 Signature and packaging validity	All signatures and container structures match their specifications across inputs and outputs.	Archive-hash mismatches, missing or swapped subfiles, malformed containers, cryptogram-format anomalies, invalid signatures or certificate usage, and structurally invalid auxiliary lists, including F_1–F_7, F_9, and F_{16}–F_{23}.
O_6 Cross-artefact consistency	Values in related artefacts match exactly, such as the registration bundle versus ballot-box contents released for auditing.	Registration-bundle hash mismatch (F_{24}), related input/list conflicts such as F_{14}, and same-district swap anomalies such as F_{31}.

3.3 Revoting secrecy under composition with eID transaction logs (Publication III)

This section reports the third contribution, which moves from internal processing correctness to a system-level secrecy property that helps motivate revoting-related processing complexity. The focus is on composition with the surrounding eID ecosystem and on how persistent transaction logs can make revoting behaviour detectable. The section summarises the adversary model, the leakage channel, and the comparative assessment of mitigation directions.

3.3.1 Threat model and the leakage channel

The third result shifts from internal processing correctness to a system-level property that partially motivates the processing-stage complexity, namely revoting-based coercion

mitigation. Publication III analyses how revoting secrecy expectations can fail under composition with the surrounding eID ecosystem, focusing on persistent transaction logs generated by remote signing and on their visibility to the voter.

The analysed coercer cannot read vote content but can pressure the voter after the election to reveal their eID transaction history. If the eID transaction log, or a user-visible view of it, exposes the number and timing of signing operations, then revoting can become detectable. The coercer does not need a receipt of the selected candidate. It can suffice to obtain a reliable signal that at least one additional voting-related signing operation occurred after the coerced session. The contribution is therefore not the mere existence of timestamps or transaction logs, but the system-composition argument that voter-visible eID history can become coercer-observable evidence about whether revoting occurred.

3.3.2 Mitigation strategies and comparative assessment

Publication III evaluates mitigation strategies along three dimensions: effectiveness, deployability, and regulatory fit. Table 4 reproduces the comparative High/Medium/Low assessment reported in the publication. These labels should be read as a relative technical assessment of mitigation directions, not as settled legal conclusions. Any deployable mitigation would have to preserve the accountability and misuse-detection functions served by eID transaction history while preventing coercer-visible inference from the existence, count, and timing of vote-related entries.

Table 4: Mitigation strategies against revoting detection via eID transaction logs (from Publication III).

Strategy	Effectiveness	Deployability	Regulatory fit
Persistent log filter (wrapper)	High	High	High
Separate voting credential	High	Low	High
Offline signing (ID card)	Medium	Medium	High
User-controlled hiding	Medium	High	High
Dummy log entries	Medium	Low	Low
Controlled log access	High	Low	High

Publication III discusses a phased response. A persistent log filter (wrapper) is presented as a near-term direction. A dedicated voting credential is framed as a longer-term structural option with higher deployment cost.

3.3.3 Implications for interpreting processing-stage complexity

The processing stage implements revoting and paper override through explicit invalidation artefacts, which increases both system complexity and the auditability burden. Publication III reframes this complexity by showing that if revoting secrecy can be undermined by persistent metadata in the eID ecosystem, then the coercion-mitigation value of revoting must be reassessed under realistic composition assumptions.

Taken together, the three publications cover artefact-level processing integrity checking, measurable audit-tool sensitivity via fault-seeded corpora and explicit oracles, and system-level constraints on revoting secrecy arising from the eID logging ecosystem. These results establish the technical and methodological basis for the synthesis and implications developed in the next chapter.

4 Discussion

This dissertation examined the auditability and security-relevant properties of Estonia's i-voting processing stage through a safety-critical testing perspective. Where Chapter 3 reported the concrete results of the three underlying publications, this chapter synthesises them into an assurance-oriented interpretation and places the findings in a broader methodological and system-architectural context.

The central theme is *testing for trust*. Trust is treated as an evidential conclusion grounded in checkable artefacts, adversarial and off-nominal tests, and explicitly stated assumptions. Across Publication I, Publication II, and Publication III, the scope expands from an artefact-level integrity oracle for processing, to a methodology for measuring audit-tool sensitivity, and finally to a system-composition constraint on revoting secrecy arising from the surrounding eID ecosystem.

The dissertation's assurance argument can be read as three linked claims. First, processing-stage integrity can be constrained by explicit, independently checkable relations over audit artefacts (Publication I). Second, the detection capability of an audit toolchain can itself be evaluated using a stated fault model, explicit oracles, and fault-seeded corpora (Publication II). Third, revoting-driven security assumptions motivating parts of the processing design can be weakened under realistic composition with external logging infrastructures (Publication III).

4.1 From isolated checks to an assurance view

This section connects the three publications into a single assurance narrative rather than treating them as independent summaries. The synthesis emphasises how the contributions relate to each other as claims, evidence surfaces, and boundary conditions.

4.1.1 An assurance trajectory across the three publications

The three publications form a coherent trajectory that separates three distinct questions arising in assurance-driven interpretations of i-voting. First, what integrity relations can be checked from audit artefacts without trusting internal execution? Second, how can auditors justify that their toolchain is sensitive to relevant deviation classes rather than merely confirming nominal operation? Third, which architectural assumptions motivate complexity, and do those assumptions continue to hold under realistic system composition?

Publication I addressed a concrete evidential gap in processing-stage audits by introducing an explicit accounting invariant over the audit artefacts. Publication II generalised from a single oracle to the *testability of the audit itself* by introducing requirement–fault mapping, fault-seeded corpora, and explicit oracles as a basis for measurable sensitivity claims. Publication III broadened the scope to system composition by showing that a major motivating security goal for revoting-related processing complexity can be weakened by persistent metadata in the surrounding eID ecosystem.

Taken together, the trajectory is not merely additive. It moves from asking whether a specific processing relation can be checked, to asking what an auditor can demonstrate about the sensitivity of the checking process itself, and finally to asking whether the surrounding architectural assumptions still justify the complexity being audited.

4.1.2 A compact claim–evidence–boundary synthesis

To avoid treating the dissertation as three independent summaries, Table 5 condenses each publication into its core claim, its evidence surface, and its boundary conditions. The purpose of this format is not only concision but also to make the structure of the

assurance argument explicit. It clarifies what is supported by audit artefacts and what requires additional assumptions or evidence sources.

Table 5: Synthesis of the three publications as claim–evidence–boundary statements.

Source	Core claim (what is shown)	Evidence surface (how it is shown)	Boundary conditions (what is not shown)
Pub. I	A checkable accounting invariant links processing inputs to anonymised outputs while making allowed invalidation paths explicit.	Checksum-normalised audit artefacts; set/balance relation (Eq. 1); nominal and manipulated scenarios evaluated by an independent script.	Union-based accounting does not prevent *semantically incorrect* category shifts; set semantics require explicit handling of duplicates and semantic constraints beyond membership.
Pub. II	Audit mechanisms can be tested as targets, enabling measurable sensitivity claims using requirement–fault mapping, fault seeding, and explicit oracles.	Model of R_1–R_{19} and F_1–F_{37}; fault-seeded exemplar corpus; explicit oracles O_1–O_6; Phase A coverage and mutation-score snapshot.	Results characterise the corpus and tool under test, not the operational system; sensitivity is bounded by the tool's evidence surface and stated scope.
Pub. III	Revoting secrecy can fail under composition with persistent eID transaction logs, weakening coercion-mitigation assumptions and reframing revoting-driven complexity.	Threat model based on post-election log disclosure; leakage via count and timing signals; mitigation comparison by effectiveness, deployability, and regulatory fit.	Practical impact depends on log visibility and coercer capabilities; mitigation feasibility depends on governance and regulatory constraints.

4.1.3 Why an integrity oracle is necessary but not sufficient

Publication I provides a concrete artefact-level oracle for processing integrity. However, an integrity oracle alone does not amount to a complete assurance argument unless auditors can also justify that the oracle is implemented correctly and that it is sensitive to the deviation classes that matter. This is a general assurance lesson for i-voting audits because the processing stage is both transformation-heavy and intentionally invalidation-heavy. A check that is correct only on nominal paths is not, by itself, a strong assurance statement in an adversarial context.

Publication II makes these concerns measurable. The Phase A results show that a log-based set-consistency checker can be effective on deviations aligned with its evidence surface while remaining insensitive to deviation classes that require different evidential hooks. This is an expected consequence of the tool's stated scope rather than a direct statement about operational system correctness. The results therefore motivate an audit approach that combines multiple independent oracles across multiple evidence surfaces.

The within-scope miss $F_{26\text{-}2}$ sharpens the point. The scenario demonstrates that an adversary can aim for a self-consistent narrative by editing the aggregates and logs on which a tool relies so that reported totals remain plausible even when per-ballot reality has changed. Methodologically, this motivates audit designs in which aggregate summaries remain auditable against per-ballot identifiers and in which assurance does not rest on a single artefact class.

Publication III explains why these methodological distinctions matter beyond internal correctness. A substantial part of processing-stage complexity exists to support revoting as a coercion-mitigation mechanism. If revoting secrecy is weakened by persistent eID metadata, then the motivating assumption must be reassessed under composition, and the role of revoting-driven complexity in the overall assurance story becomes less straightforward.

4.2 Testing for trust: conceptual implications

This section clarifies what "testing for trust" means as a general assurance stance and how it relates to established verifiability concepts. The goal is to articulate conceptual takeaways that are broader than any single IVXV artefact or tool version.

4.2.1 Trust as a conclusion rather than a premise

The term *trust* is used here in a deliberately narrow assurance sense. It refers to the evidential warrant for relying on specific processing and auditability claims, not to the broader sociological or human-centred question of how citizens form trust in election institutions, interfaces, or authorities. Therefore, "testing for trust" should be read as testing for warranted assurance claims, not as a claim that improved auditing automatically increases public trust.

The dissertation adopts the software-independence stance that outcome-relevant deviations must leave detectable traces in artefacts and results [27]. In this view, assurance should not rest on trusting software execution, procedural correctness, or institutional intent. Instead, assurance rests on checkable relations that independent parties can evaluate from the audit artefacts. The dissertation adds a methodological bridge by treating the *audit process itself* as a test target and by asking what deviation classes the audit can actually detect under a stated fault model.

This reframes common audit outcomes. A statement such as "no anomalies were found" is meaningful only relative to explicit assumptions and to demonstrated sensitivity of the audit procedure with respect to the deviation classes considered. Without that sensitivity argument, "no anomalies" may merely mean "no anomalies of the kind this procedure can see."

4.2.2 An asymmetry between correctness assurance and privacy assurance

The dissertation also points to an asymmetry between correctness assurance and privacy assurance. For result correctness, the methodological aim is to minimise dependence on procedural trust by shifting assurance toward retrospective verification from released artefacts, explicit oracles, and independently reproducible checks. For privacy and coercion resistance, however, the situation is less symmetric. Some guarantees still depend on constrained disclosure, correct handling of revoting-related data, and the behaviour of surrounding infrastructures that cannot be reduced to the same artefact-based audit model without changing the secrecy assumptions themselves.

This asymmetry helps explain why the dissertation's first two contributions primarily strengthen assurance of outcome correctness, while the third contribution reframes the limits of secrecy assurance under realistic system composition. In other words, the work suggests that software-independent auditability can currently be pushed further for correctness than for privacy, at least within the present architectural setting.

4.2.3 Verifiability evidence is broader than cryptographic proofs

In the i-voting literature, verifiability is often discussed through cryptographic mechanisms and formal proofs, especially in the context of end-to-end (E2E) verifiability [1, 2].

In the specific case of IVXV, the literature itself no longer supports a single unqualified E2E-verifiability claim. Earlier work presented IVXV as satisfying E2E-verifiability requirements, whereas later work in the same research line argued that the current Estonian Internet voting scheme should be understood as balancing verifiability against secrecy and coercion-resistance constraints rather than as satisfying the strongest interpretations of E2E verifiability outright [14, 13]. The present dissertation builds on this shift by focusing not on whether IVXV is E2E-verifiable in the abstract, but on what can be independently justified from released artefacts, validated audit procedures, and explicit evidence surfaces.

In practice, real elections necessarily combine cryptographic assurances with procedures, released evidence, and auditor-driven evaluation. From this perspective, the artefact-level invariants and explicit oracles used in this dissertation can be understood as a complementary form of verifiability evidence. They are not cryptographic zero-knowledge proofs, but they are machine-checkable, precisely specified predicates that constrain what outcomes are compatible with the audit artefacts under explicit assumptions. A system may therefore incorporate E2E-verifiability mechanisms while still leaving open practical questions about whether critical transformations are independently checkable in deployed audits, whether the audit toolchain's sensitivity has been demonstrated, and how narrow the verifier community remains in practice. Even under stronger verifiability architectures, not every practically relevant election decision becomes fully public or purely cryptographic. Eligibility determination, voter-register management, and validity-related decisions arising from revoting or paper-ballot override may still depend on controlled institutional evidence rather than on a publicly checkable cryptographic proof alone.

This matters for how IVXV assurance claims are interpreted. The processing stage is a point where cryptographic secrecy constraints and operational invalidation rules intersect, and where auditors must reason about transformations rather than about a single cryptographic transcript. In such settings, artefact-level invariants and cross-artefact constraints function as a practical bridge from protocol claims to audit evidence. The dissertation therefore contributes to the E2E discussion by showing how precisely specified, checkable integrity relations over audit artefacts can strengthen software-independent assurance even when the check is not itself a cryptographic proof.

4.2.4 Evidence surfaces, explicit oracles, and semantic constraints

A central concept in Publication II is the *explicit oracle*. An oracle is a deterministic predicate over audit artefacts that defines whether a deviation should be detected in a given test case. This shifts auditing from procedure-driven judgement toward reproducible evaluation with measurable outcomes. Publication I provides an oracle in the form of an accounting invariant, while Publication II generalises this into a family of oracles capturing different semantic constraints.

The distinction between accounting and semantics is essential. Publication I makes explicit that union-style accounting does not prevent category shifts between valid and invalid sets when the union is preserved. It also makes explicit that set semantics require explicit handling of duplicates and multiplicity when duplicates are a plausible deviation. These boundaries motivate treating semantic properties as first-class oracles rather than as informal expectations. Examples include last-vote correctness under revoting semantics, correct application of paper override, and preservation of district-related attributes through transformations.

4.3 Implications for audit practice and system governance

This section translates the dissertation's results into implications for how audits are organised, validated, and reported. The emphasis is on reducing reliance on procedural confidence and increasing reliance on validated detection capability.

4.3.1 Pre-election validation of audit toolchains

Publication II implies that audit validation should not begin only after an election. A safety-critical interpretation suggests maintaining a regression-style fault-seeded corpus with explicit oracles, enabling repeated validation of audit tools across election cycles. Such validation answers a different question than post-election audit execution. It asks what deviation classes the toolchain can detect under the stated fault model, and what blind spots remain.

This is particularly important when organisers provide auditing tools. Organiser-provided tools may still be useful, but auditors should not rely on them without independent sensitivity evaluation. A validated corpus and oracle framework enables auditors to justify tool use while retaining independence at the level of evidence and test outcomes.

4.3.2 Who audits matters as much as how auditing is done

The dissertation's methodology also highlights that auditability is not only a technical property but also an institutional one. The ability to construct fault models, define oracles, generate mutants, and interpret sensitivity results depends on auditor competence and independence. In practice, "testing for trust" therefore implies governance requirements: credible independence from the system implementers, sufficient technical capacity to validate tools, and procedures that allow audits to be repeated and reproduced.

This perspective connects directly to public credibility. If the audit process relies on expertise that cannot be independently exercised or repeated, then the audit result may remain persuasive only within a narrow expert group. Conversely, publishing corpora, oracles, and validation results can widen the set of parties who can reproduce and critique assurance claims. Such transparency reduces the risk that trust shifts from evidence to a tool merely because the tool is official or convenient.

Some international cases illustrate the governance point. In specific jurisdictions, i-voting deployments have faced sustained scrutiny and, in some cases, termination or redesign decisions after reassessing assurance and trust arguments [21, 24]. In an assurance-oriented reading, such episodes underline that trust is not only a technical claim but also an institutional outcome shaped by what evidence is provided, how it is validated, and who can reproduce it.

4.3.3 From pass/fail outcomes to sensitivity and coverage reporting

Binary audit outcomes are information-poor summaries. A more informative reporting structure aligns with Publication II and expresses assurance in terms of sensitivity and coverage. A report can state the explicit scope and assumptions, the covered requirement and fault classes, the detection outcomes on fault-seeded variants, and the documented blind spots. Such reporting helps prevent "no anomalies" from being interpreted as "no plausible anomalies."

4.3.4 Normative clarity and implicit policy discovered in code

The requirement–fault mapping exercise in Publication II also revealed a governance risk that is easy to miss in purely technical accounts. Code may implement a policy decision even when that policy is not stated as a requirement in documentation or in the normative

framework understood by stakeholders. One recurring example in the dissertation context concerns how anomalous situations such as multiple ballots with identical cryptogram checksums are handled. If such cases are treated as revoting-like events rather than as invalid inputs, then the system effectively embeds a normative choice about how anomalies are to be interpreted.

This matters because anomaly-handling policy is too consequential to remain an implicit developer decision. If an anomaly is a strong indicator of error or manipulation, then treating it as a normal case can mask the presence of a fault and weaken audit interpretability. From a safety- and mission-critical assurance perspective, frameworks such as NASA's IV&V [20] emphasise defensive checks and error handling that preserve or return the system to a safe state, including rejection of invalid or unsafe inputs rather than reliance on best-effort interpretation. Accordingly, the findings of this dissertation highlight the value of making anomaly-handling rules explicit in requirements and clarifying how such cases are treated within the assurance model, for example through rejection, quarantine, or a separately auditable exceptional procedure.

This theoretical concern was illustrated operationally in the 2025 local elections audit. The audit report describes a case where repeated submission of the same cryptogram by a voter, using a self-made voter application, triggered an anomaly in the auditor application's cryptogram check because the tool did not account for a usage case in which the same cryptogram appears both among counted e-votes and among invalidated e-votes [23]. The report further notes that the official voter application does not permit repeated submission of the same cryptogram in normal operation [23]. This episode does not change the formal results of Publication I or Publication II. Rather, it reinforces the practical need to specify the intended semantics for duplicate or replay-like artefacts explicitly and to validate audit tooling against realistic input variability beyond the official client's behaviour [23].

In principle, if a system component were to accept replay of an already seen cryptogram *despite* voter-identity binding controls, for example under a failure mode or interface misuse, then the assurance implications could extend beyond repeated submissions by a single voter. This is not claimed to occur in IVXV. It illustrates why intended semantics must be explicit and testable: the meaning of "duplicate" depends on which binding assumptions are treated as invariants and which are treated as contingent on implementation correctness.

4.3.5 Governance implications of the eID logging ecosystem

Publication III highlights a composition tension between the voting system and the surrounding trust-service ecosystem. The voting system seeks to avoid persistent evidence of revoting behaviour, while trust-service infrastructure is designed for durable audit trails and non-repudiation. Accordingly, mitigation is not purely technical. It depends on governance decisions about what transaction history is exposed to voters, how it is presented, and what institutional controls exist over access and disclosure.

The publication discusses a phased response, with wrapper-style filtering of voter-visible log views presented as a near-term direction and dedicated voting credentials presented as a longer-term structural option. From an assurance viewpoint, this reinforces that coercion mitigation cannot be treated as an isolated protocol property. It depends on the behaviour and visibility of the surrounding identity and logging infrastructures.

4.4 Generalisability and significance

This section clarifies what aspects of the dissertation generalise beyond IVXV and why the contributions are significant scientifically and practically. The aim is to separate system-specific artefacts from more general patterns.

4.4.1 What generalises are patterns rather than IVXV artefacts

Although the dissertation is instantiated in IVXV, three patterns are likely to generalise. The first is balance-style integrity invariants, where a process that intentionally discards or invalidates inputs can be audited by binding the input to the output plus explicit removal classes. The second is treating audits as test targets and evaluating tools using fault seeding, explicit oracles, and measurable sensitivity outcomes. The third is composition-aware security interpretation, where security goals such as coercion mitigation can fail through metadata leakage even when cryptographic protections remain intact.

What does not generalise directly are IVXV-specific artefact formats and pipeline semantics. Other systems would require re-modelling their artefacts, invalidation rules, and evidence surfaces before the same patterns can be instantiated. Nevertheless, the dissertation provides a method for carrying out that re-modelling as part of an explicit assurance argument. For example, a system without revoting or paper-ballot override would not need the same invalidation classes, but it would still need to state which input artefacts may legitimately disappear from the counted set and where that decision is evidenced. A system built around stronger cryptographic transcripts may shift part of the assurance burden from administrative artefacts to public proofs, but it would still need explicit oracles for eligibility, validity, and operational decisions that remain outside the transcript. Likewise, systems using return codes, public bulletin boards, or different identity infrastructures would instantiate different evidence surfaces, while preserving the methodological requirement that claimed assurance be tied to checkable evidence and stated fault classes.

4.4.2 A policy fork and its effect on the technical design space

The revoting analysis also exposes a strategic choice that affects both security claims and engineering complexity. If revoting secrecy is treated as essential for coercion mitigation, then leakage channels such as voter-visible transaction logs become assurance-critical and must be addressed. Publication III does not imply that this leakage problem is unfixable. It shows, however, that under the present system composition revoting secrecy can be weakened by metadata flows in the surrounding eID ecosystem.

The dissertation therefore does not argue that revoting should be abandoned. It argues that the rationale for parts of the revoting-driven processing complexity depends on whether the corresponding secrecy assumptions are actually protected in practice. If the leakage channel is mitigated, then that rationale can in principle remain intact. If it is not mitigated, then the trade-off between secrecy constraints and audit complexity must be reassessed against the system as it exists in practice rather than against the election protocol in isolation.

4.5 Limitations and future work

This section states the main boundaries under which the results should be interpreted and outlines the most direct extensions. The aim is to be explicit about what is not covered and what would strengthen the evidence base.

4.5.1 Limitations

First, the dissertation focuses on the processing stage and its artefacts and does not provide a full client-side security analysis or attempt to cover all network- and infrastructure-level threats. Second, the sensitivity evaluation in Publication II is based on synthetic corpora rather than live election artefacts. This is both a limitation and a methodological choice because synthetic corpora enable controlled fault seeding and explicit oracle evaluation,

but cannot capture all operational variance. Third, the composition analysis in Publication III assumes an adversary who can pressure a voter to disclose eID transaction history and interpret the log view. The analysis is best understood as identifying a plausible and structurally grounded leakage channel rather than as an empirical estimate of prevalence.

4.5.2 Future work

Several directions follow naturally from the dissertation results. A direct extension is to apply the same methodology to later stages such as mixing, counting, and publication of results by defining explicit oracles and fault models for those stages and by building corresponding fault-seeded corpora. A second direction is to strengthen semantic constraints beyond set accounting by defining additional validity-classification predicates and evaluating them under fault-seeded scenarios. This direction addresses the category-shift boundary of union-style accounting and supports stronger processing assurance within broader end-to-end verifiability goals. A third direction is to automate corpus generation and oracle evaluation so as to reduce human error and improve repeatability across election cycles. A fourth direction is to explore governance- and deployment-feasible mitigations for eID-log leakage, including wrapper-style filtering of voter-visible transaction history and longer-term structural options such as dedicated voting credentials. Finally, controlled operational pilots that validate audit toolchains against realistic workflows under appropriate privacy constraints would strengthen the evidence base and support institutional adoption.

5 Conclusion

This chapter draws together the main conclusions of the dissertation and answers the three research questions stated in Chapter 1. It highlights the principal findings, their implications for audit practice and governance, and the broader assurance stance developed in this work. The central theme is *testing for trust*, in which trust is treated as an evidential conclusion grounded in checkable artefacts, explicit assumptions, and validated detection capability.

5.1 Answers to the research questions

This section provides concise answers to the three research questions. The answers emphasise both what is concluded and the evidence surface that makes each conclusion checkable.

5.1.1 RQ1: How can processing-stage integrity be verified from artefacts without trusting the internal logic of the processing application?

The dissertation showed that processing-stage integrity can be audited using an artefact-based accounting invariant that binds the signed input ballot box to the anonymised output while making allowed invalidation paths explicit. The invariant separates technical invalidity from rule-based invalidations and expresses integrity as a balance relation over checksum-normalised audit artefacts. Under this model, every ballot that is not technically invalid must be accounted for either as a counted anonymised output ballot or as an explicitly invalidated ballot in the revoting or paper-override artefacts. This yields a machine-checkable deviation signal for unexplained additions, removals, or replacements across processing-stage artefacts. At the same time, the dissertation clarified the boundaries of such checks, including the need to handle duplicates explicitly and the fact that union-style accounting alone does not prevent semantically incorrect category shifts.

5.1.2 RQ2: How can safety-critical testing principles be applied to evaluate the quality and sensitivity of auditing tools?

The dissertation demonstrated that the evidential power of an audit toolchain can be evaluated by treating auditing itself as a test target and by applying fault seeding with explicit test oracles. A structured model of functional requirements and fault scenarios enables systematic construction of synthetic test corpora that include deliberate deviations paired with deterministic oracle predicates over audit artefacts. This supports measurable sensitivity claims, such as coverage of modelled deviation classes and mutation-style detection rates for specific tools under test. The reported Phase A snapshot in the exemplar corpus illustrates that detection capability is bounded by a tool's evidence surface and stated scope, and that this can be demonstrated rather than assumed. Accordingly, "no anomalies found" is meaningful only relative to a stated fault model and demonstrated audit sensitivity, which in turn motivates combining multiple independent oracles and evidence surfaces to reduce blind spots.

5.1.3 RQ3: How do eID transaction logs affect revoting secrecy and coercion-mitigation assumptions?

The dissertation analysed revoting secrecy under composition with the national eID ecosystem and showed that persistent transaction logs can create a metadata side channel for revoting detection. In the considered threat model, a coercer who cannot observe ballot content may still infer the occurrence and timing of revoting if voter-visible transaction

history reveals the number or timing of signing operations. This weakens the practical assumption that revoting behaviour is not reliably inferable or externally observable after the election, which underlies revoting as a coercion-mitigation mechanism in remote settings. The analysis therefore reframes revoting-driven processing complexity as dependent on system-level information flows beyond the IVXV boundary. Mitigation directions were compared, highlighting near-term measures that reduce voter-visible log signals and longer-term structural options that separate voting-related signing traces from ordinary eID transaction history, subject to governance and regulatory constraints.

5.2 Implications for practice and governance

This section summarises implications that follow from the dissertation results. The emphasis is on strengthening audit credibility by making scope, sensitivity, and assumptions explicit and reproducible. These implications concern auditability and assurance arguments, not a claim that operational election outcomes were incorrect.

5.2.1 Pre-election validation using fault-seeded corpora

The results imply that audit preparation benefits from deliberate negative-scenario testing in addition to nominal checks. Maintaining a regression-style corpus of fault-seeded datasets with explicit oracles provides a repeatable baseline for validating that audit tools detect the deviation classes they claim to detect. Such validation answers a different question than operational post-election auditing. It characterises tool sensitivity under a stated fault model and clarifies which blind spots remain before tools are relied upon operationally.

5.2.2 Stating tool scope and evidence surfaces explicitly

These findings also imply that organiser-provided tools can support assurance only if their sensitivity boundaries are known and validated against an explicit fault model. Audit reporting can therefore benefit from stating, for each tool, which deviation classes are in scope, which audit artefacts and evidence surfaces the tool relies on, and which risks remain out of scope. Where feasible, using independent evidence surfaces reduces the risk that assurance depends on a single self-consistent narrative.

5.2.3 Composition-aware interpretation of coercion mitigation

A further implication is that, if revoting is relied upon as a coercion-mitigation mechanism, the surrounding eID logging ecosystem becomes assurance-relevant. Reducing voter-visible transaction-log leakage can therefore be understood as part of maintaining the intended coercion-mitigation assumptions under realistic system composition. Near-term directions include wrapper-style filtering of voter-visible log views, whereas longer-term options include structural separation such as dedicated voting credentials, subject to governance feasibility and regulatory constraints.

5.3 Closing remarks

The dissertation supports an assurance stance in which durable trust in a high-stakes sociotechnical system is better grounded in evidence than presumed through procedure. For IVXV processing, such evidence begins with artefact-level integrity relations that account explicitly for allowed invalidations and transformations. For audit practice, it extends to the audit itself, where sensitivity to modelled deviation classes can be validated and communicated transparently. At the system level, it must also include composition effects,

because security goals such as coercion mitigation may be constrained by metadata leakage outside the voting-system boundary.

The dissertation should therefore not be read as settling, in the abstract, whether IVXV satisfies end-to-end verifiability as a scheme-level property. Rather, it clarifies what parts of the assurance argument can be independently grounded in released artefacts, validated audit procedures, and explicit evidence surfaces, and what parts remain dependent on controlled institutional evidence, constrained disclosure, or surrounding infrastructures. In that sense, the dissertation complements broader verifiability claims by making the operational assurance boundaries more explicit.

A central conclusion of the dissertation is that result correctness and ballot secrecy do not appear to admit identical assurance models within the present architectural setting. For correctness, the methodological goal should be to replace procedural trust with retrospective, artefact-based verification as far as possible. For secrecy and coercion resistance, some dependence on correct procedure, constrained disclosure, and infrastructure behaviour remains unavoidable unless the underlying system design is changed.

By moving from confirming nominal operation to validating detection capability, *testing for trust* supports assurance conclusions that are clearer, more reproducible, and better aligned with the realities of election-system auditing across election cycles.

List of Figures

1 The steps and dataflow of the processing stage (Check–Squash–Revoke–Anonymise), including classes of invalidated votes, reproduced from Publication I and originally adapted from [30]. 21
2 Transforming ballots for comparison (reproduced from Publication I). 25

List of Tables

1 Thesis map from research questions to approach and publications. 18
2 Phase A coverage and sensitivity snapshot (from Publication II). 26
3 Representative requirement–fault mapping patterns organised by oracles (summary view based on Publication II). 27
4 Mitigation strategies against revoting detection via eID transaction logs (from Publication III). 28
5 Synthesis of the three publications as claim–evidence–boundary statements. 30
6 Full fault catalogue used in the Phase A methodology. 109
7 Functional requirements consolidated for the processing application in Publication II. 111
8 Dissertation-level requirement–fault mapping used for Phase A coverage planning. 112
9 Fault scenarios evaluated primarily through system-level invariants rather than a single explicit documented requirement. 113

References

[1] J. Benaloh, R. Rivest, P. Y. A. Ryan, P. Stark, V. Teague, and P. Vora. End-to-end verifiability. *CoRR*, abs/1504.03778, Apr. 2015. arXiv:1504.03778 [cs].

[2] M. Bernhard, J. Benaloh, J. A. Halderman, R. L. Rivest, P. Y. A. Ryan, P. B. Stark, V. Teague, P. L. Vora, and D. S. Wallach. Public evidence from secret ballots. In *Electronic Voting*, Lecture Notes in Computer Science, pages 84–109. Springer International Publishing, 2017.

[3] D. Clarke and S. T. Ali. End to end security is not enough. In *Security Protocols XXV*, pages 260–267. Springer International Publishing, 2017.

[4] S. Dzieduszycka-Suinat, J. Murray, J. R. Kiniry, D. M. Zimmerman, D. Wagner, P. Robinson, A. Foltzer, and S. Morina. The future of voting: End-to-end verifiable internet voting – specification and feasibility assessment study. Technical report, U.S. Vote Foundation and Galois, July 2015.

[5] P. Ehin, M. Solvak, J. Willemson, and P. Vinkel. Internet voting in Estonia 2005–2019: Evidence from eleven elections. *Government Information Quarterly*, 39(4):101718, Oct. 2022.

[6] ERR. Riigikohus lükkas kõik valimiskaebused tagasi. `https://www.err.ee/1608931658/riigikohus-lukkas-koik-valimiskaebused-tagasi`, Mar. 2023.

[7] ERR. EKRE complaint calls for e-voting to be stopped immediately. `https://news.err.ee/1609830015/ekre-complaint-calls-for-e-voting-to-be-stopped-immediately`, Feb. 2025.

[8] European Parliament and Council. Regulation (EU) no 910/2014 of the European Parliament and of the council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market. `https://eur-lex.europa.eu/eli/reg/2014/910`, 2014.

[9] European Telecommunications Standards Institute. ETSI EN 319 401 v3.1.1: Electronic Signatures and Infrastructures (ESI); General Policy Requirements for Trust Service Providers. `https://www.etsi.org/deliver/etsi_en/319400_319499/319401/03.01.01_60/en_319401v030101p.pdf`, 2024.

[10] K. Fowler. Mission-critical and safety-critical development. *IEEE Instrumentation & Measurement Magazine*, 7(4):52–59, 2004.

[11] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 265–273, New York, NY, USA, 2008. Association for Computing Machinery.

[12] T. Haines, S. J. Lewis, O. Pereira, and V. Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 644–660, 2020.

[13] S. Heiberg, K. Krips, and J. Willemson. Planning the next steps for Estonian internet voting. *E-Vote-ID 2020*, pages 82–97, 2020.

[14] S. Heiberg, T. Martens, P. Vinkel, and J. Willemson. Improving the verifiability of the Estonian internet voting scheme. In *Electronic voting*, Lecture notes in computer science, pages 92–107. Springer International Publishing, 2017.

[15] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 61–70, New York, NY, USA, 2005. Association for Computing Machinery.

[16] K. Krips and J. Willemson. On practical aspects of coercion-resistant remote voting systems. In *Electronic Voting*, pages 216–232, Cham, 2019. Springer International Publishing.

[17] P. A. Laplante and J. F. DeFranco. Software engineering of safety-critical systems: Themes from practitioners. *IEEE Transactions on Reliability*, 66(3):825–836, 2017.

[18] M. McGaley and J. P. Gibson. Electronic voting: A safety critical system. *Final Year Project Report, NUI Maynooth Department of Computer Science*, 2003.

[19] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, and B. Monate. Testing or formal verification: DO-178C alternatives and industrial experience. *IEEE Software*, 30(3):50–57, 2013.

[20] NASA. Software assurance and software safety standard. Technical Report NASA-STD-8739.8B, National Aeronautics and Space Administration (NASA), Sept. 2022. Revision B.

[21] Norwegian Ministry of Local Government and Modernisation. Internet voting pilot to be discontinued. `https://www.regjeringen.no/en/historical-archive/solbergs-government/Ministries/kmd/press-releases/2014/Internet-voting-pilot-to-be-discontinued/id764300/`, 2014.

[22] A. J. Offutt. A practical system for mutation testing: help for the common programmer. In *Proceedings., International Test Conference*, pages 824–830, 1994.

[23] J. Oruaas and H. Põldmaa. Kohaliku omavalitsuse volikogu valimiste 2025 elektroonilise hääletuse toimingute audit: Lõpparuanne. Technical report, FocusIT OÜ, Dec. 2025.

[24] O. Pereira and V. Teague. Report on the SwissPost-Scytl e-voting system, trusted-server version. `https://www.bk.admin.ch/dam/bk/de/dokumente/pore/E-Voting_Report_Pereira_Teague_Juli%202019.pdf.download.pdf/E-Voting_Report_Pereira_Teague_Juli%202019.pdf`, July 2019.

[25] Rapita Systems Ltd. Efficient verification through the DO-178C life cycle. White paper MC-WP-011, v4, Rapita Systems Ltd., Dec. 2021.

[26] Riigikogu. Riigikogu valimise seadus. `https://www.riigiteataja.ee/akt/124052024010`, 2024. English translation available.

[27] R. L. Rivest. 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, Aug. 2008.

[28] P. Sewell and J. Vitek. Secure composition of insecure components. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 136–150, 1999.

[29] SK ID Solutions AS. Principles of processing personal data (privacy policy). https://www.skidsolutions.eu/upload/files/SK-POPPD-EN-CURRENT.pdf, 2022.

[30] State Electoral Office of Estonia. Elektroonilise hääletamise üldraamistik ja selle kasutamine Eesti riiklikel valimistel. https://www.valimised.ee/sites/default/files/2023-02/IVXV%20elektroonilise%20h%C3%A4%C3%A4letamise%20%C3%BCldraamistik.pdf, Feb. 2023.

[31] State Electoral Office of Estonia. E-hääletamise auditeerimine saab koostöös teadlastega taas täienduse. https://www.valimised.ee/et/e-haaletamise-auditeerimine-saab-koostoos-teadlastega-taas-taienduse, Dec. 2025. News post published 09.12.2025.

[32] State Electoral Office of Estonia. IVXV: E-hääletamise käsiraamat. https://www.valimised.ee/sites/default/files/2025-09/RVT%20korraldus%20nr%2012%20lisa%20IVXV%20e-h%C3%A4%C3%A4letamise%20k%C3%A4siraamat.pdf, Sept. 2025.

[33] State Electoral Office of Estonia. IVXV online voting system. https://github.com/valimised/ivxv, 2025.

[34] State Electoral Office of Estonia. IVXV protokollide kirjeldus. https://www.valimised.ee/sites/default/files/2025-09/RVT%20korraldus%20nr%2013%20lisa%20IVXV%20protokollide%20kirjeldus.pdf, Sept. 2025.

[35] State Electoral Office of Estonia. Statistics about Internet voting in Estonia. https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia, Dec. 2025.

[36] Supreme Court of Estonia. Riigikohtu põhiseaduslikkuse järelevalve kolleegiumi otsus 3-4-1-13-05. https://www.riigikohus.ee/et/lahendid?asjaNr=3-4-1-13-05, 2005. English translation available.

[37] T. Treier. Beyond the happy path: A safety-critical audit methodology for Estonia's i-voting processing application. *IEEE Access*, 13:203429–203443, 2025.

[38] T. Treier. Re-voting under surveillance: National eID transaction logs as a threat to coercion resistance in Estonian internet voting. In D. Duenas-Cid, P. Roenne, M. Volkamer, M. Blom, P. Gaudry, I. Borucki, L. Loeber, and A. Debant, editors, *Electronic Voting*, pages 191–207, Cham, 2025. Springer Nature Switzerland.

[39] T. Treier and K. Düüna. Identifying and solving a vulnerability in the Estonian internet voting process: Subverting ballot integrity without detection. *IEEE Access*, 12:197766–197782, 2024.

[40] P. Vinkel. Internet voting: Experiences from five elections in Estonia. *Democracy and Development—Taiwan and Baltic Countries in Comparative*, Apr. 2012.

[41] L. Wang and K. C. Tan. Software testing for safety critical applications. *IEEE Instrumentation & Measurement Magazine*, 8(2):38–47, 2005.

Acknowledgements

I would first like to express my deepest gratitude to my supervisor, Innar Liiv. When I returned to doctoral studies after an earlier interrupted start and with a research topic that was not directly within his main field of research, he nevertheless agreed to support me. More than detailed subject-matter guidance, I needed someone who would believe in me, trust my direction, and help me navigate the long path to completion with patience and encouragement. Innar fulfilled that role exceptionally well. His trust, support, and timely motivation were invaluable throughout this journey.

I am also grateful to the State Electoral Office of Estonia for constructive cooperation during the course of this research. Their willingness to review my observations and improvement proposals at early stages of the work helped me maintain focus and gave me confidence that I was addressing questions of real practical relevance.

My deepest thanks go to my family. Above all, I thank my wife, Jana, who has supported this work for years. She kept alive the belief that I should finish this doctorate and made the writing phase possible by taking on more than her share of everyday responsibilities. I am profoundly grateful for her patience, encouragement, and constant support.

I also thank my children, Trevor, Kendra, Trevis, and Keron. They are my greatest source of joy and motivation. Their simple questions about how the dissertation was progressing often meant more than they could know. I hope that, in completing this work, I can show them that perseverance matters and that even goals that may seem out of reach for a long time can be achieved.

Finally, I wish to acknowledge the use of AI-based tools during the preparation of this dissertation for language and editorial support. All research questions, methodological choices, analyses, interpretations, and conclusions presented in this dissertation are my own.

Abstract

Testing for Trust: Safety-Critical Auditability and Limits of Coercion Resistance in the Estonian Internet Voting System

Internet voting (i-voting) increases voter convenience, but it concentrates two difficult and partly competing requirements: independently checkable election integrity and strong ballot secrecy in an uncontrolled environment. Estonia's long-running nationwide deployment makes these tensions operational rather than purely theoretical. This dissertation addresses a concrete auditability gap in Estonia's IVXV system at the server-side *processing stage*, where voter-linked, signed containers carrying encrypted ballots are filtered under revoting and paper-override rules and then anonymised prior to tallying. Because the processed output is designed to be unlinkable, processing forms an evidential bottleneck. It is the last stage at which an independent auditor can relate outcome-relevant transformations to voter-linked inputs without relying on trusted internal execution.

The dissertation develops an evidence-based assurance stance summarised as *testing for trust*. It treats trust as an evidential conclusion, in which claims are expressed as checkable relations over released election artefacts, under explicit assumptions, and are validated against adversarial and off-nominal scenarios. The dissertation is structured around three linked research questions and corresponding contributions. It progresses from an artefact-level integrity oracle for processing, to a safety-critical-inspired methodology for validating audit-tool sensitivity, and finally to a system-level composition analysis showing how revoting secrecy assumptions are constrained by the surrounding national eID ecosystem.

First, the dissertation formulates a checkable integrity relation for processing that can be evaluated from audit artefacts without trusting the processing application's internal logic. The core idea is an accounting invariant that links the signed input e-vote collection to the anonymised output e-ballot collection while making all protocol-prescribed invalidation paths explicit and independently checkable. Using checksum-normalised representations of heterogeneous artefacts, the invariant requires that every input ballot that is not technically invalid is accounted for either as an anonymised output ballot or as an explicitly invalidated ballot under revoting or paper-ballot override. The dissertation also delineates the boundaries of this approach. Union-style accounting alone does not prevent semantically incorrect category shifts, and set-based checks require explicit handling of duplicate and multiplicity corner cases where relevant.

Second, the dissertation shows that defining an integrity oracle is necessary but not sufficient for a credible audit argument. Auditors must also justify that the audit toolchain is sensitive to the deviation classes that matter. To this end, the dissertation introduces a methodology that treats auditing tools and procedures as test targets. Based on documentation and source-code analysis, the work models functional requirements and fault scenarios, constructs fault-seeded test datasets, and defines explicit deterministic audit oracles as predicates over the evidence surface. This enables measurable sensitivity claims about what classes of deviations a given audit toolchain can and cannot detect under stated assumptions. It also motivates multi-oracle audits and transparent reporting of blind spots.

Third, the dissertation analyses a system-level constraint that reshapes how revoting-driven complexity and related audit assumptions should be interpreted. Estonia relies on revoting as a practical coercion-mitigation mechanism, while the broader national eID ecosystem supports durable transaction logging. The dissertation shows that, under realistic system composition, persistent eID transaction logs can create a metadata side channel.

A coercer who cannot observe ballot content may still infer whether revoting occurred by examining counts and timing patterns in voter-visible transaction-history records disclosed after the election. The work compares mitigation strategies by effectiveness, deployability, and regulatory fit, highlighting near-term directions such as filtering voter-visible transaction-history records and longer-term structural options such as separate voting credentials, subject to governance and legal constraints.

Overall, the dissertation contributes an auditability-oriented framework in which processing integrity is constrained by explicitly checkable artefact relations, audit toolchains are validated through systematic fault modelling, fault seeding, and explicit oracles, and revoting secrecy is interpreted compositionally by accounting for external metadata flows. Practically, the work supports more objective and repeatable auditing across election cycles by shifting emphasis from confirming nominal operation to validating detection capability under stated assumptions and explicit definitions of audit artefacts and evidence surfaces. More broadly, it shows that result correctness and privacy do not admit the same assurance model: result correctness can be supported through retrospective, artefact-based verification, whereas privacy and coercion resistance remain partly dependent on constrained disclosure and the behaviour of surrounding infrastructures.

Kokkuvõte
Usaldus läbi testimise: ohutuskriitiline auditeeritavus ja sundimiskindluse piirid Eesti internetihääletussüsteemis

Internetihääletamine (i-hääletamine) suurendab valija mugavust, kuid koondab kaks keerukalt täidetavat ja osaliselt vastandlikku nõuet: valimistulemus peab olema sõltumatult kontrollitav ning samal ajal tuleb säilitada tugev häälesaladus kontrollimatus keskkonnas. Eesti pikaajaline üleriigiline i-hääletuse kasutus muudab need pinged praktiliseks, mitte pelgalt teoreetiliseks. Käesolev doktoritöö käsitleb Eesti IVXV süsteemis konkreetset auditeeritavuse lünka serveripoolses *töötlemise etapis*, kus valijaga seotud, allkirjastatud konteinerid, mis sisaldavad krüpteeritud sedeleid, filtreeritakse kordushääletuse ja paberhääle ülimuslikkuse reeglite järgi ning seejärel anonümiseeritakse enne häälte lugemist. Kuna töötlemise väljund on kavandatult valijaga mitteseostatav, kujutab töötlemine endast tõenduslikku kitsaskohta. See on viimane punkt, kus sõltumatu audiitor saab tulemust mõjutavaid teisendusi seostada valijaga seotud sisendartefaktidega ilma töötlemisrakenduse sisemist täitmist usaldamata.

Doktoritöö arendab välja tõenduspõhise lähenemise, mida kokkuvõtlikult kirjeldab mõiste *usaldus läbi testimise*. Selles käsitluses ei ole usaldus eeldus, mis tugineb peamiselt protseduuridele ja usaldatud täitmisele, vaid järeldus, mis põhineb kontrollitavatel artefaktidel, selgelt sõnastatud eeldustel ning ründe- ja veastsenaariume hõlmaval valideerimisel. Töö on üles ehitatud kolme omavahel seotud uurimisküsimuse ja neile vastava panuse ümber. See liigub töötlemisetapi artefaktitaseme terviklusoraaklist edasi ohutuskriitilistest valdkondadest inspireeritud metoodikani audiitortööriistade avastamisvõime valideerimiseks ning lõpuks süsteemse kompositsiooni analüüsini, mis näitab, kuidas kordushääletuse saladuse eeldusi piirab ümbritsev riiklik eID ökosüsteem.

Esiteks sõnastab töö kontrollitava terviklusseose töötlemise jaoks, mida saab hinnata audiitorile kättesaadavate artefaktide põhjal ilma töötlemisrakenduse sisemist loogikat usaldamata. Tuumidee on bilansiseos, mis seob allkirjastatud sisendsedelite kogumi anonümiseeritud väljundsedelite kogumiga ning teeb kõik protokolliga lubatud kehtetuks tunnistamise teed selgelt eristatavaks ja sõltumatult kontrollitavaks. Heterogeensete artefaktide võrdlemiseks kasutatakse kontrollsummadel põhinevat normaliseerimist. Bilansiseos nõuab, et iga tehniliselt kehtiv sisendsedel oleks kajastatud kas anonümiseeritud väljundsedelina või selgesõnaliselt kehtetuks tunnistatud sedelina kordushääletuse või paberhääle ülimuslikkuse tõttu. Töö toob esile ka lähenemise piirid. Pelgalt hulkade ühendil põhinev arvestus ei välista semantiliselt valesid klassivahetusi ning hulgapõhised kontrollid eeldavad asjakohasel juhul duplikaatide ja paljususe erijuhtude eraldi käsitlemist.

Teiseks näitab töö, et tervikluskontrolli oraakli määratlemine on vajalik, kuid mitte piisav usaldusväärse auditeerimisargumendi jaoks. Audiitor peab suutma põhjendada, et kasutatav auditi tööriistaahel on tundlik just nende hälbeklasside suhtes, mis on ohumudeli mõttes olulised. Selleks esitab töö metoodika, mis käsitleb auditeerimist ennast testimise sihtmärgina. Dokumentatsiooni ja lähtekoodi analüüsi põhjal modelleeritakse funktsionaalsed nõuded ning vea- ja ründestsenaariumid, koostatakse sihilikult vigadega rikastatud sünteetilised testandmestikud ning määratletakse selgesõnalised deterministlikud auditi oraaklid kui kontrolltingimused tõenduspinnal. See võimaldab esitada mõõdetavaid väiteid selle kohta, milliseid hälbeklasse konkreetne auditiprotsess või tööriist suudab või ei suuda avastada selgesõnaliselt esitatud eelduste korral. Samuti motiveerib see mitme sõltumatu oraakli kasutamist ja pimedate tsoonide läbipaistvat raporteerimist.

Kolmandaks analüüsib töö süsteemset piirangut, mis mõjutab kordushääletusest tuleneva keerukuse ja sellega seotud auditieelduste tõlgendamist. Eestis kasutatakse kordus-

hääletust praktilise sundimise leevendusmehhanismina, kuid laiem riiklik eID ökosüsteem toetab püsivat tehingulogimist. Töö näitab, et realistliku süsteemse kompositsiooni korral võivad eID tehingulogid moodustada metaandmetel põhineva kõrvalkanali. Sundija, kes ei näe hääle sisu, võib siiski järeldada kordushääletuse toimumist, kui valijale nähtav tehinguajalugu paljastab allkirjastamistoimingute arvu või nende ajamustrid. Leevendusstrateegiaid võrreldakse tõhususe, rakendatavuse ja regulatiivse sobivuse lõikes, tuues esile nii lühiajalisi suundi, näiteks valijale nähtava logivaate filtreerimise, kui ka pikaajalisi struktuurseid lahendusi, näiteks eraldi hääletamiseks mõeldud autentimisvahendid, arvestades juhtimis- ja õiguslikke piiranguid.

Kokkuvõttes annab doktoritöö auditeeritavusele suunatud raamistiku, milles töötlemisetapi terviklust piiravad selgelt kontrollitavad artefaktiseosed, auditi tööriistaahelaid valideeritakse süsteemse veamudeli, sihilikult vigadega rikastatud testandmestike ning selgesõnaliste oraaklite abil ning kordushääletuse saladust tõlgendatakse kompositsiooniliselt, arvestades väliseid metaandmevooge. Praktiliselt toetab töö objektiivsemat ja korratavamat auditeerimist valimistsüklite lõikes, nihutades fookuse nominaalse toimimise kinnitamiselt avastamisvõime valideerimisele selgelt sõnastatud eelduste ning auditiartefaktide ja tõenduspinna täpsete määratluste raames. Laiemas plaanis näitab töö, et valimistulemuse korrektsus ja privaatsus ei allu samale usaldusmudelile: korrektsuse tagamisel saab toetuda tagantjärele läbiviidavale artefaktipõhisele tõendamisele, samas kui privaatsus ja sundimiskindlus jäävad osaliselt sõltuma piiratud avalikustamisest ning ümbritseva taristu käitumisest.

Appendix 1

I

T. Treier and K. Düüna. Identifying and solving a vulnerability in the Estonian internet voting process: Subverting ballot integrity without detection. *IEEE Access*, 12:197766–197782, 2024

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 November 2024, accepted 19 December 2024, date of publication 23 December 2024,
date of current version 31 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3521337

RESEARCH ARTICLE

Identifying and Solving a Vulnerability in the Estonian Internet Voting Process: Subverting Ballot Integrity Without Detection

TARVO TREIER AND KRISTJAN DÜÜNA
Department of Software Science, Tallinn University of Technology, 12618 Tallinn, Estonia
Corresponding author: Tarvo Treier (tarvo.treier@taltech.ee)

ABSTRACT In Estonia, nationwide internet voting (i-voting) has been in use since 2005, with its popularity steadily rising. In the recent parliamentary election in March 2023, over half of the voters cast their ballots electronically. This study focuses on the security of i-voting, specifically potential vulnerabilities to insider attacks and the integrity of the i-ballot-box. We reviewed the i-voting source code, analyzed the operational system in our lab, studied documentation, and examined audit reports from previous elections. Our systematic examination of the vote processing stage revealed vulnerabilities that could allow a dishonest insider to replace all ballots undetected. To address this, we proposed an audit application to verify that no ballots have been altered during the processing stage. The formula was rigorously tested across multiple scenarios, including ballot replacement, addition, and removal, proving its capability to detect a comprehensive range of manipulations. This study also explores the historical development of the i-voting system, end-to-end verifiability, and over-the-shoulder coercion-resistance. It highlights the limitations of existing coercion-resistant systems and the need for further scrutiny. Our findings suggested additional measures to ensure i-ballot-box integrity. The proposed methodology offers a framework for auditors to enhance voting process security, contributing to the reliability and transparency of internet voting systems. Notably, the source code for the European Parliament election, published on May 30, 2024, includes an enhanced auditing application based on our script.

INDEX TERMS Internet voting, ballot box integrity, end-to-end verifiability, coercion-resistance, transparency, auditing.

I. INTRODUCTION

In 2005, Estonia became the world's first country to implement remote internet voting (i-voting) nationwide [1]. From the beginning, alongside i-voting, it has been possible for everyone to vote with a paper ballot at the polling station. While other countries experimented with i-voting on a limited scale or opted not to continue because of concerns about privacy and integrity in general [2], Estonia emerged as a pioneer, allowing i-voting in all subsequent nationwide elections. The usage of i-voting was initially modest, with less than 2% of all voters opting for it in the first election that offered this option. However, in the recent Estonian parliamentary election in March 2023, over 50% of all voters chose electronic voting [3]. This shift underscores the growing significance of i-voting in Estonia's electoral landscape.

The discourse surrounding the security of Estonian i-voting has been intense, with debates and criticism leading to calls for its annulment. These calls have not been heeded. Critics have been met with the argument that the source code related to i-voting is publicly accessible, and everyone who has registered beforehand can observe i-voting and verify its reliability themselves. Furthermore, i-voting processes undergo international audits conducted by accredited audit firms, which have attested to the absence of significant

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Kheir.

deviations from established standards, enhancing the system's credibility.

The authors of this research have chosen to heed the wisdom of an old Russian proverb famously quoted by President Ronald Reagan during the height of the Cold War in the 1980s: "Doveryai, no proveryai" (translated as "Trust but verify"). In diplomatic relations, trusting other nations and their promises is crucial, but blind trust is ill-advised. Similarly, trust is pivotal in the context of i-voting; however, it must be rational and verifiable. While numerous aspects of i-voting security demand scrutiny, this study primarily focuses on the hypothetical risk of dishonest insider attacks and the integrity of the i-ballot-box on the central server, where the processing of ballots occurs after the voting period is concluded. In this study, integrity refers specifically to ensuring that no ballots are added or removed from the ballot box. In exploring these risks, we also highlight the importance of ensuring verifiability and integrity through systematic validation of the processes involved in maintaining the i-ballot-box.

The main objective of this study is to ascertain whether dishonest insiders could manipulate ballots without being detected by observers or auditors and to propose solutions to the identified issues. This research delves into the core of i-voting security, aiming to ensure the reliability and integrity of the electoral process, a cornerstone of any robust democracy.

A. METHODOLOGY

This study employs a mixed methodology, which combines a thorough review of i-voting source code [4], analysis of the operational i-voting system within our laboratory environment, and examination of documentations [5], [6], [7], along with the study of an audit report from the recent Estonian parliamentary election [8]. The methodology is broken down into the following elements:

- Review of i-voting source code: Thoroughly reviewing the source code of the i-voting system to understand the implementation and identify potential vulnerabilities.
- Analysis of the operational i-voting system: Studying the operational i-voting system within a controlled laboratory environment to gain practical insights into its functioning and potential vulnerabilities.
- Examination of documentation: Studying relevant documentation related to the i-voting system, including reports, guidelines, and specifications, to understand the system's design and security measures.
- Study of audit reports: Analyzing audit reports from previous elections to assess the effectiveness of the auditing process and identify any detected deviations or vulnerabilities.
- Responsible disclosure: Engaging in a cooperative dialogue with the representatives of the Estonian State Electoral Service to address identified vulnerabilities and submitting a script for additional auditing of the i-voting process.
- Proposal of an audit application: Proposing an audit application to enhance the verification of the i-ballot-box integrity during the processing stage.

We systematically scrutinized every step of the vote processing stage based on the source code, raising the question at each step of how and if the system detects any unauthorized addition, removal, or substitution of ballots. To ensure the robustness of our findings, we validated the proposed integrity formula against a range of test scenarios, including normal and manipulated cases.

B. RESPONSIBLE DISCLOSURE

We formally notified the representatives of the Estonian State Electoral Service about our discovery in writing on March 30, 2023. The representatives were cooperative and expressed interest in addressing the identified vulnerability. They provided us with the ballot-box, logs, and configurations from a test election they conducted for our analysis.

On April 12, 2023, we submitted a Python script to the State Electoral Office for additional auditing of the i-voting process. This script enabled a detailed examination of the i-voting logs to detect any manipulation of the ballots during the processing stage.

On December 5, 2023, the representatives confirmed that our provided script was accurate and contributed significantly to increasing the transparency of the i-voting process. Subsequently, on May 30, 2024, the source code for the European Parliament election was published, incorporating an enhanced auditing application based on our submitted script and current study.

II. EVOLUTION OF ESTONIAN INTERNET VOTING

The following sections provide an in-depth exploration of the evolution of internet voting in Estonia, a pioneering nation in the field of digital democracy.

The 'Historical Development' subsection delves into the early years of Estonian i-voting, highlighting foundational reports that shaped its inception and addressing challenges such as internal attacks and technological limitations. 'End-to-End Verifiability' discusses the concept and its implementation in the Estonian context, while the 'Ongoing Challenges' subsection offers an overview of current obstacles faced by the Estonian i-voting system.

This comprehensive examination sheds light on the complexities of implementing and maintaining a secure and verifiable internet voting system, emphasizing the importance of continuous evaluation and improvement in the face of evolving technological landscapes and security threats. The insights gained from Estonia's experience can serve as a valuable guide for nations venturing into the field of digital democracy.

A. HISTORICAL DEVELOPMENT

Estonian i-voting commenced in the early 2000s with nationally commissioned reports [9] and [10], exploring

implementation possibilities and providing recommendations for future considerations. From the very first report [9], an internal attack orchestrated by compromised election organizers was envisaged as a potential threat, underscoring the importance of ensuring verifiability in all steps of the i-voting system. Proposing a solution, [9] suggested that all political parties should have the capability to perform security audits on servers and could deploy their independent servers. The second report [10] proposed integrity verification by suggesting control codes derived from the cast vote transmitted to the server, which would be provided to the voter as a response after voting. These control codes could then be collected from voters after the elections in a specific sample, and their validity could be compared with the codes stored on the server. As of the elections conducted so far, voters have not been provided with the aforementioned control codes, and political parties have not been granted access to conduct security audits.

In 2003, the architect of the Estonian election information system, Tarvi Martens, published a conceptual solution [11], based on which the first local government election were planned to be conducted online in 2005. The concept aimed to ensure the trustworthiness of central servers through workflow and auditor control. The concept mentioned the need to securely ensure the integrity of the vote collection during the transition from one stage to another but did not specify how to achieve this. Additionally, it stated that the audit application should be capable of checking the integrity of logs by comparing logs obtained at different stages. In the previous solution used in the last parliamentary election in March 2023, each output of a stage was signed and its conformity was verified in the next stage. In this study, we demonstrate that ensuring ballot box integrity requires not only checking the transition process from one stage to another but also proving that there has been no manipulation of ballots during a single stage. Unlike the described concept, the audit application used in the last parliamentary election in March 2023 did not enable the verification of log integrity. Even if logs were compared by someone at some point, auditors did not confirm this in the audits of the recent Estonian parliamentary election.

In a retrospective on the world's first nationwide internet voting [1], National Electoral Committee member Ülle Madise and election information system architect Tarvi Martens acknowledged that meeting the security requirements of i-voting, namely confidentiality and auditability, simultaneously is challenging due to their conflicting nature. Since the first occurrence, the solution in use has been the "digital double envelope" concept, analogous to the solution for postal voting, where an anonymous ballot is placed in an internal envelope, and the external envelope contains the voter's information. The external envelope is removed when they reach the polling station, and the anonymous ballot is added to the ballot box. According to the authors of this study, there is a crucial difference in terms of privacy between postal voting and the solution used in internet voting. In postal voting, once the external envelope is separated from the internal envelope, reassembling them is extremely difficult. While theoretically possible using advanced methods like invisible marking, handwriting analysis, or DNA testing, it remains highly impractical. By contrast, in internet voting, with the existence of logs and copies of the double envelope, it is relatively easy to do so. While we consider confidentiality important in this study, and our proposed solutions must not compromise the existing state, our main focus is on investigating ballot box integrity and its auditability.

B. END-TO-END VERIFIABILITY

The pamphlet created as a result of the Overseas Vote Foundation's project, "End-to-End Verifiable Internet Voting: Specification and Feasibility Assessment Study" [12], explains to a non-technical audience what end-to-end verifiability (E2E-V) is and how it can be achieved. In an end-to-end verifiable election, individual voters should be able to check crucial elements of election results without having to trust the election software, hardware, election officials, procedures, or even observers. E2E-V is closely associated with software independence [13], which states that a voting system is software independent if any change or error in its software, whether detected or undetected, cannot result in an undetectable change or error in the outcome of an election. In this study, we also strive for E2E-V, with the acknowledgment that, after the voter ensures the integrity of their ballot in the ballot box, they may rely on checks by observers and auditors in the subsequent process, similar to traditional paper-based voting.

In 2016, researchers involved in the development of Estonian i-voting published improvements in the verifiability of the i-voting scheme [14]. These enhancements represented major updates focused on increasing the verifiability of processes taking place on central servers, and they were also used in the recent Estonian parliamentary election in March 2023. They introduced measures to ensure the auditability of the correctness of vote decryption and i-ballot-box integrity. In summary, they claimed that they achieved E2E-V. In a security analysis [15], published in 2021, it was also shown that E2E-V has been achieved in practice. While this was a significant step in the right direction, in this study, we demonstrate that additional audit steps are necessary to prove i-ballot-box integrity. While the referenced articles addressed the entire election process, our focus is a more in-depth examination of only the i-ballot-box processing stage, which deals with the removal of invalid ballots and the anonymization of valid ballots.

C. ONGOING CHALLENGES

Jan Willemson, a researcher associated with the development of Estonian i-voting, published an article [16], which reviewed several dimensions for comparing internet and paper voting, including vote secrecy, verifiability, ballot box integrity, transparency, and trust base. This article was partly

a response to the security analysis by Springall et al. [17], which criticized an older version of Estonia's i-voting system and recommended discontinuing its use. Based on their tests, they concluded that state-level attackers, sophisticated criminals, or dishonest insiders could manipulate election outcomes by circumventing both technological and procedural controls. J. Willemson concluded in [16] that many vulnerabilities of i-voting systems have related weaknesses in paper systems as well, and instead of attacking i-voting, the focus should be on maximizing security by implementing strong cryptographic authentication tokens, enhancing digital ballot box integrity, and developing verifiability techniques.

In 2019, the Estonian Minister of Foreign Trade and Information Technology, Kert Kingo, convened a working group to assess the compliance of the i-voting system and i-voting information system processes and security measures with existing regulations on cybersecurity and election organization. The working group included representatives from government institutions, academic institutions, and individuals. The working group submitted a report [18], that contained 25 proposals regarding the security of the system and raising public awareness. In this study, we adhere to at least two of these proposals, namely, the recommendation to enhance the post-control procedure for the software components of the i-voting information system and to strive for E2E-V.

After the introduction of i-voting in Estonia, the Office for Democratic Institutions and Human Rights (OSCE/ODIHR) has observed parliamentary elections five times. In their latest report [19], they were generally satisfied with Estonia's i-voting system, as the majority of their previous recommendations had been taken into account. However, they observed that the critical step of removing votes overwritten by another vote cast on the internet or in a polling station was not audited. In addition, they reminded in the report of the previous recommendation to consider methods to achieve E2E-V. To achieve this, they suggested addressing issues with individual verifiability and ensuring that all critical steps in determining the results of i-voting are auditable.

In a recent review [20] examining the auditability of the same version of the Estonian internet voting system as addressed in this study, a previously unknown vulnerability undermining individual verifiability was identified. The review suggested that due to the system's lack of adequate specification, this vulnerability went undetected, leading to the inference that the system likely harbors additional vulnerabilities of a similar nature.

Throughout the review process, the compliance of the Estonian internet voting system with the New Standards for E-Voting Systems [21], previously published by two co-authors of the review, was evaluated. It was determined that only one out of the nine criteria from the standard, minimal restrictions on disclosure, was satisfied. For example, they were not able to build the system as a whole and therefore could only test particular functionalities of the system.

While the review was confined to English-language documents, our study benefited from the utilization of Estonian-language materials. Nevertheless, we concur with the reviewers' assessment that constructing and effectively operationalizing the entire system solely based on publicly available information is practically unattainable. For testing the stage evaluated in this study, we also needed to rely on configuration examples obtained through direct communication with representatives of the Estonian State Electoral Service, in addition to publicly available documentation and source code.

III. ESTONIAN INTERNET VOTING PROTOCOL IVXV

In this section, we briefly explore the entire i-voting IVXV protocol [7], the fundamental process of which has been in use since 2017. Specifically, we delve into the part of the process that occurs between the end of the voting period and the counting of votes, examining the mechanisms that ensure the integrity of the i-ballot-box in this segment.

The IVXV i-voting framework [6] divides i-voting into four stages:

1) Pre-voting Stage: The system is prepared for use, involving the compilation of district, polling station, candidate, and voter lists. Encryption and decryption keys are generated for each vote, and voter and verification applications, along with relevant instructional materials, are published. The data necessary for authenticity and integrity checks are also disclosed.
2) Voting Stage: I-voting takes place, potentially simultaneously in polling stations during parallel voting.
3) Processing Stage: The integrity and authenticity of electronic votes (e-votes) are verified. Votes are sorted, and any repeated e-votes by the same person are annulled, and only the most recent vote is retained. In cases of parallel voting, a list of double voters is generated. Next, the e-votes of double voters, who have cast both an e-vote and a paper ballot, are annulled accordingly. Finally, the remaining votes to be counted are anonymized.
4) Counting Stage: Anonymized votes are opened with a decryption key and counted to determine the election outcome.

The components of the IVXV protocol can be categorized according to their usage in different stages, as depicted in FIGURE 1. According to the protocol, voters use the Voter Application during the voting stage to electronically formulate their preferences in three steps: firstly, the expression of preferences is transformed into an e-vote; secondly, the formulated e-vote is packed into an encrypted ballot using a key generated by the Key Application, and finally, it is digitally signed. The Collector Service stores the signed ballot, taking into account the validity of the voter's certificate, and registers the ballot in the Registration Service. The voter is provided with the option to use the Verification Application designed for checking qualified

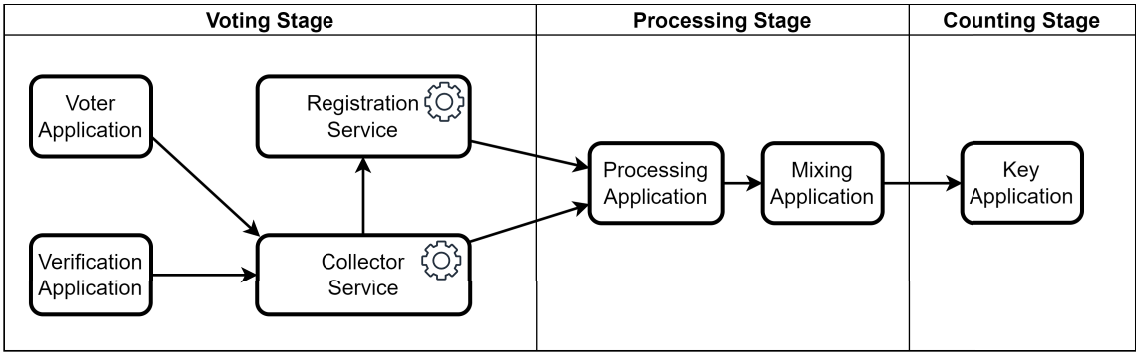

FIGURE 1. The components of the IVXV protocol.

electronic votes. At the conclusion of the voting stage, the Collector Service issues the i-ballot-box, and the Registration Service generates a list of ballots registered by the Collector Service. In the processing stage, the Processing Application is used to identify valid ballots, anonymize them by separating ballots and digital signatures, and, optionally, use the Mixing Application, which alters the ballot's ciphertext, making it unlinkable to the original ballot while keeping the encrypted content intact. During the counting stage, the election organizer calculates the election outcome using the Key Application to decrypt anonymous and encrypted ballots and computes the outcome based on the decrypted ballots.

In this work, our focus shifts exclusively to the processing stage and the integrity of the i-ballot-box during this stage.

A. PROCESSING STAGE

The processing stage is conducted on an offline computer and consists of four steps, as shown in FIGURE 2, and described according to the IVXV i-voting framework [6]:

- Check: Verify the integrity and authenticity (digital signature) of e-votes, ensuring that all given e-votes are still present.
- Squash: Sort e-votes; duplicate e-votes from the same person are revoked and only the newest one is retained. Additionally, improperly encrypted e-votes are revoked.
- Revoke: In the case of parallel voting, the election information system receives the list of individuals who have voted electronically. Subsequently, it identifies individuals who have also cast paper ballots, resulting in a list of double voters. The electronic votes of these individuals are then invalidated.
- Anonymize: E-votes ready for counting are anonymized.
- Mix (optional): Anonymized votes undergo a mixing process and are then re-encrypted, altering their ciphertext, the content remains unchanged upon decryption.

As input, the processing stage receives the i-ballot-box issued by the Collector Service, and the output is a ballot box containing the remaining anonymous ballots along with district information. If mixing was used, the mixed votes cannot be directly associated with previous anonymous votes. The correctness of mixing can be verified by auditors using

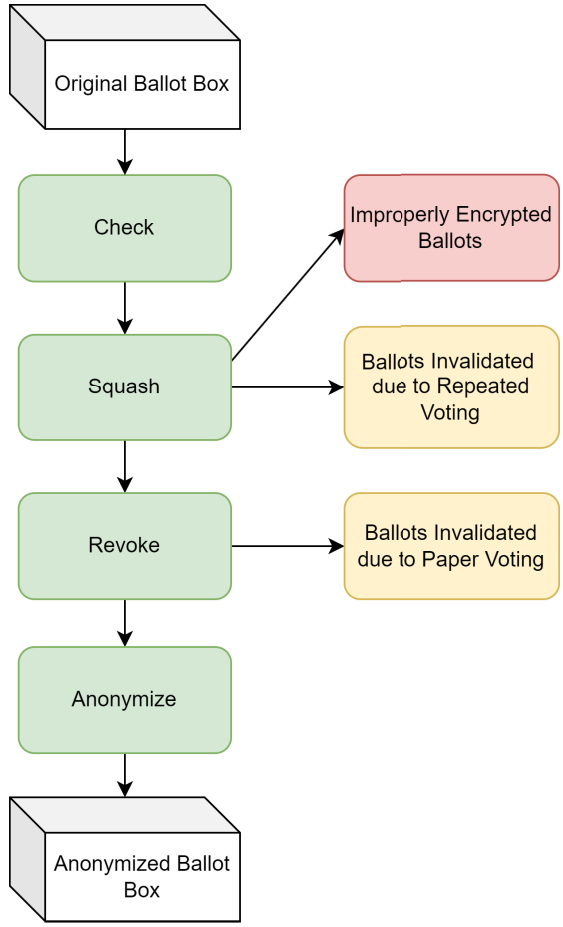

FIGURE 2. The steps of the processing stage.

an auditor application and the mixing proof, which is a byproduct of the mixing step. Because mixing is optional and does not affect the current study into the integrity of the i-ballot-box, the discussion on the mixing step is omitted from further consideration for the sake of simplicity.

B. SECURITY FEATURES OF BALLOT AND ITS VULNERABILITY DURING THE ANONYMIZATION

When the processing stage begins, each ballot in the i-ballot-box is protected by the following security features:

- Public-Key Encryption: The ballot submitted by the voter includes the code of the choice in the district, the code of the district, the name of the choice list, and the name of the specific choice in the list. This information, along with a random number, is encrypted using the public key provided by the election organizer. The inclusion of a random number ensures that each resulting cryptogram is unique, even for voters with the same choice. The primary purpose of public-key encryption is to maintain the confidentiality of the voter's intent until the votes are counted.
- Voter's Signature: The ballot encrypted with the public key is signed by the voter. The signature confirms that it is a specific voter's ballot.
- Timestamp: All signed ballots have a timestamp confirming that at the time of making the request, the ballot existed.

In the first step of the processing stage, the correctness of all three security elements is checked. By the end of the processing stage, the signature and timestamp are removed. Only the ballots protected by public- key encryption remain in the ballot box. While this encryption protects the secrecy of the vote, it does not protect against forgery during anonymization. Anyone can use the public information to create a correctly formatted ballot encrypted with the public key. However, not everyone can replace ballots because they are processed on an offline computer. This does not exclude the possibility that dishonest insiders could replace ballots during anonymization with seemingly correct ones, which, when decrypted during the counting stage, would contain a correctly formatted ballot but with a different choice than the original.

In the next section, we will examine how the system ensures that no ballots are replaced during the processing stage in the ballot box.

C. AUDITING I-BALLOT-BOX INTEGRITY IN THE PROCESSING STAGE

To ensure adherence to the valid guidelines [5] of i-voting and assess data integrity, an audit is commissioned. In the recent Estonian parliamentary election in March 2023, KPMG Baltics OÜ conducted the audit.

In the processing stage, all input data file checksums, obtained through the standard sha256sum command, are signed, and auditors verify the signed checksum's conformity to the actual input file checksum. This verification is confirmed in the audit report [8]. Additionally, the i-ballot-box issued as a result of each step of the processing stage is signed.

The IVXV i-voting framework [6] allows auditors to verify all processing stage steps through re-execution. According to the actions carried out in the audit report [8], the processing steps have not been repeated by auditors.

Upon communicating with the Estonian State Electoral Service, it was revealed that auditors conduct a sampling check manually during the processing stage. However, neither the audit report [8] nor the guidelines [5], [6], [7] confirm or explain the exact content and scope of this sampling check.

In an introductory article on the IVXV protocol [14], similar to the i-voting framework [6], it is stated about the verifiability of the processing stage that the steps are well-defined and repeatable, meaning that the process consistently produces the same outputs for the same inputs regardless of implementation. Furthermore, the article suggests that through the complete re-execution of the processing stage steps, it is possible to perform risk-limiting audits. This term likely refers to the aforementioned sampling check conducted manually. During the sampling check, there must be a reason for revocation for each vote that is removed from the i-ballot box during the processing stage. According to multiple sources (see, e.g., [22], [23]), the risk-limiting audit method requires paper ballots as the source of truth. The Estonian internet voting system does not produce voter-verifiable paper records; however, according to the authors of this study, this method could still be applied, as digitally signed ballots by the voter could serve as a reliable alternative to paper ballots.

IV. ASSESSING THE INTEGRITY OF THE I-BALLOT-BOX

In this section, we delve into the critical issue of assessing the integrity of the i-ballot-box during its processing stage. We critically examine the current auditing measures and those proposed thus far and discuss why, in the authors' view, are insufficient to ensure the integrity of the i-ballot-box. We also explore the challenges that these measures present and discuss what aspects need further scrutiny to ensure the i-ballot-box's integrity.

A. I-BALLOT-BOX SIGNING

In the context outlined earlier, it is crucial to understand that the checksum of the input i-ballot-box is signed at each processing step, allowing auditors to verify its consistency with the actual checksum of the file. When the i-ballot-box serves as the input for a particular step, the checksum validation assures us that it remains unaltered from the i-ballot-box generated in the preceding step. However, it is essential to note that the checksum of the initial i-ballot-box loses its relevance in evaluating the integrity of the resulting i-ballot-box. This is because these two checksums are no longer comparable once the content of the i-ballot-box

undergoes modifications. Each processing step introduces changes to the i-ballot-box content, necessitated by the requirement to invalidate specific ballots or to remove voter details. Consequently, the checksum of the input i-ballot-box for steps such as anonymization cannot and should not be identical to the checksum of the i-ballot-box produced as the output.

B. MANUAL SAMPLING CHECK

The practical application of the concept of manual sampling checks in Estonian i-voting remains unclear for the authors. Nevertheless, we posit that it involves a meticulous verification process. This process entails the random selection of ballots from the original i-ballot-box, followed by an examination of their presence in the anonymized ballot box. Should these selected ballots not be found in the anonymized set, the scrutiny extends to determining whether they are duly listed as invalid, accompanied by a thorough investigation into the validity of the reasons for their cancellation. Similar checks can be conducted in the reverse direction as well. This process entails randomly selecting a ballot from the pool of anonymized ballots and verifying its origin by ensuring it matches with the initial ballot box. The rationale behind such meticulous checks lies in their ability to provide robust statistical evidence, affirming the accuracy of the output generated by the processing steps. However, even with a large and diverse sample, a sampling check cannot guarantee absolute certainty.

Illustrating the challenge, recent i-voting in Estonia for the latest parliamentary election saw a significant voter turnout, with over three hundred thousand votes cast [3]. Undertaking a manual review of such a substantial volume requires a considerable time and resource investment. Complicating matters further are the data that undergo transformations throughout the processing steps. The i-ballot-box houses files with the .ballot extension. These files represent ballots in binary form. The i-ballot-box is then transformed into the anonymized ballot box in JSON format. In the JSON format, ballots are encoded in base64.

Moreover, insights from international observers, particularly OSCE/ODIHR, underscore a critical aspect—that the step involving the removal of votes overwritten by another vote, either online or at a polling station, lacked a comprehensive audit [19]. This observation implies that conducting extensive manual sampling checks at this scale might pose practical challenges. As mentioned in the article [24], even an end-to-end verifiable voting system that creates sufficient evidence to reveal problems is not enough on its own. Evidence must be used to verify correctness.

C. RE-EXECUTION OF PROCESSING STEPS

Given that the processing steps are intended to produce consistent results for the same input, an auditor's re-execution of these steps serves as a means of validating result consistency. However, mere result alignment does not assure the integrity of the i-ballot-box. If both organizers and auditors employ a malfunctioning or manipulated software program for executing these steps, they could generate similarly inaccurate results. Auditors can claim correctness through comprehensive examination and testing of the software they employ. Despite well-documented tasks for the processing steps, and the public availability of the software's source code [4], ensuring the accuracy of this software remains a challenging endeavor. The source code is crafted with multithreading and abstractions, introducing complexity for non-professional programmers to comprehend the algorithm. While re-execution of the processing steps alongside prior testing by auditors offers a potential solution, conducting thorough testing of the software based on its source code might prove to be a complex task in this context.

D. ENSURING I-BALLOT-BOX INTEGRITY: KEY CHECKS

At the outset of the processing steps, all ballots are consolidated within a single i-ballot-box. As the processing concludes, the ballots are categorized into two groups: valid and invalid. To ascertain the integrity of the i-ballot-box during processing, a crucial verification involves demonstrating that the total, obtained by combining the counts of valid and invalid ballots, aligns with the original number of ballots within the i-ballot-box.

A similar equality check for sets is outlined in the article [25], which establishes the performance requirements for E2E-V elections. The integrity check of the i-ballot-box aligns with one of the performance requirements: the consistency check, which ensures a secure chain of custody from the voters to the talliers. In other words, it verifies that the set of ballots collected during the voting from voters is the same as the set of ballots used for tallying the election results. In this study, we employ the consistency check to evaluate the input and output of the Processing Application.

Beyond a mere numerical count, an essential facet of ensuring integrity necessitates a meticulous examination of the ballots themselves. This scrutiny is imperative because the ballots, lacking inherent security features against manipulation, are vulnerable to tampering. This approach seeks to guarantee not only the numerical accuracy of the processed ballots but also the inviolability of their content.

While the hash values of valid votes are conveniently accessible in anonymized form within a JSON file, details concerning invalid votes are spread across multiple log files, referencing the votes, and checksums generated through the sha256sum command. Notably, the actual hash value of the vote is not logged. The primary challenge lies in the practical implementation of ensuring that the combination of these two sets produces a third set identical to the predefined one. This complexity arises because the sets are not initially presented in a comparable form.

The authors contend that this complexity could be the primary reason why implementing such integrity checks has been challenging and has not been done so far. From

the perspective of the final voting result, it is pivotal to verify the accuracy of categorizing votes as valid or invalid during processing. However, this study exclusively concentrates on integrity checks, signifying that no ballots have been tampered with—added, removed, or altered—during processing.

V. OVER-THE-SHOULDER COERCION-RESISTANCE

In this chapter, we examine why the processing stage is necessary and explore how other end-to-end verifiable i-voting systems handle the tasks of the processing stage and ensure the integrity of the i-ballot box.

End-to-end verifiable i-voting systems have evolved significantly over the past decade, each addressing unique challenges in ensuring voter privacy and election integrity. This chapter aims to position the Estonian i-voting system within this broader landscape by comparing it with other notable systems.

The systems selected for comparison in this chapter were initially chosen based on their relevance to end-to-end verifiability. However, during the course of this research, it became evident that coercion-resistance is a critical aspect that must also be considered. Therefore, the analysis was expanded to include systems that specifically address coercion-resistance, providing a more comprehensive evaluation.

One of the earliest and most widely cited internet-based end-to-end verifiable voting systems is Helios [26], published in 2008. When comparing the Helios system to Estonia's i-voting system, the pre-voting and voting stages are noticeably different. However, in both systems, the result at the end of the voting period is encrypted ballots. Helios seems to have skipped the primary steps of the processing stage, as it moves directly to mixing, decrypting, and tallying votes. Both Helios and the Estonian system allow auditors to verify that mixing, decryption, and tallying were done correctly using similar proofs.

The main reason for the processing stage in the Estonian system is to allow re-voting with a new e-vote and paper ballots. The re-voting option is used as a countermeasure against potential coercion, whereas Helios was explicitly designed for low-coercion elections such as student government or local clubs.

As stated in [27], a coercion-resistant voting system allows the user to deceive the adversary into thinking they have behaved as instructed while the voter has cast a ballot according to their own intentions. To make Helios coercion-resistant like the Estonian i-voting system, one could add the option for re-voting similar to the Estonian solution. This would enable the use of the same i-ballot box integrity verification method proposed in this article.

Several advancements have been made based on Helios. For instance, Apollo [28] made verification of mixnet integrity more efficient, and another advancement [29] replaced the bulletin board concept with decentralized storage and a blockchain. Neither of these advancements has made Helios coercion-resistant.

In the same year as Helios, Civitas [30] was published as a coercion-resistant voting system. According to [30], it was not yet suitable for national elections but used fake credentials and re-voting to counter coercion. If a voter felt coerced, they could cast a vote using fake credentials, making it appear correct to the coercer. While fake credentials can effectively counter coercion, they pose challenges such as complexity in creation and the risk of voters forgetting or confusing them. These challenges are beyond the scope of this article. In Civitas, voters can re-vote, and it includes steps similar to the Estonian processing stage to remove duplicates and invalid ballots, making valid ballots anonymous. However, like the Estonian system, proofs are available that results were derived from anonymous ballots, but there are no checks that the anonymous votes originate from the ballot box confirmed at the end of the voting period.

Selections [31], published in 2012, is another coercion-resistant internet voting system that uses a panic password instead of fake credentials to create an invalid ballot that appears correct but is later removed. Selections also allows re-voting and, similar to previously discussed systems, anonymizes ballots so they cannot be matched to the original ones. A distributed group of trustees handles the separation of valid and invalid ballots and their anonymization. We believe that similar integrity verification methods to those used in Estonia could be applied in Selections due to the similar challenges.

There are several other end-to-end verifiable internet voting systems designed for low-coercion elections like Helios, including more recent articles such as those by Kiayias et al. [32], Bistarelli et al. [33], Zhang et al. [34], and Boyen et al. [35]. Additionally, some systems claim to be coercion-resistant, such as those described by Shakiba et al. [36] and Kumar et al. [37], but we argue they are not fully coercion-resistant. In these systems, voters receive a receipt that allows them to verify if their vote is included in the tally, but such a receipt does not prove how they voted. However, if a coercer can observe the voter during voting and see the receipt for a coerced vote, they can be assured that the coerced vote was counted. Therefore, no voting system that provides a receipt allowing direct verification of the ballot's presence in the tally can be considered over-the-shoulder coercion-resistant.

Selene [38] is an end-to-end verifiable voting scheme introduced in 2016, which aimed to mitigate coercion by offering receipt-freeness. However, Selene lacks other countermeasures, such as the option for re-voting, which is crucial for addressing over-the-shoulder coercion. As a result, despite its strengths, Selene remains vulnerable to this type of coercion. TrustedEVoting (TeV) [39], published in 2019, does offer the ability to re-vote. However, TeV still falls short of being fully resistant to over-the-shoulder coercion, as it relies on a blockchain-based voting process. In this system, even though re-voting is allowed, the coercer could potentially

TABLE 1. Comparison of End-to-End Verifiable I-Voting Systems.

System	Year	Key Features	Coercion-Resistance	References
Helios	2008	Web-based open-audit voting, simple implementation	No, designed for low-coercion elections	[26]
Civitas	2008	Fake credentials, re-voting, complex credential management	Yes, uses fake credentials and re-voting	[30]
Apollo	2016	Efficient mixnet verification, improved performance	No, focuses on mixnet efficiency	[28]
Improved Helios	2018	Decentralized storage, blockchain integration	No, not coercion-resistant	[29]
Selections	2012	Panic password, distributed trustees	Yes, uses panic password and re-voting	[31]
End-to-End Verifiable Elections	2015	Standard model, robust cryptographic proofs	No, focuses on standard model	[32]
End-to-End Voting with Ledgers	2019	Non-permissioned ledgers, ledger-based verification	No, not coercion-resistant	[33]
Chaintegrity	2020	Large-scale e-voting, blockchain scalability	No, focuses on scalability and universal verifiability	[34]
Epoque	2021	Post-quantum security, advanced cryptographic techniques	No, focuses on quantum security	[35]
ESIV	2017	Identity-based blind signature, secure internet voting	No, not fully coercion-resistant	[36]
Secure End-to-End Verifiable Internet-Voting	2020	Blind signature, identity-based encryption	No, not fully coercion-resistant	[37]
Selene	2016	Transparent verifiability, receipt-freeness	No, lacks re-voting option	[38]
TrustedEVoting (TeV)	2019	Blockchain-based voting, verifiable re-voting	No, re-voting detectable by coercer	[39]
VoteAgain	2020	Scalable, coercion-resistant, deterministic ballot filling	Yes, uses re-voting	[40]

detect that the voter has re-voted, thereby undermining the effectiveness of this defense.

The VoteAgain scheme, published in 2020 [40], along with its extensions proposed in 2023 [41], shares several similarities with the voting system used in Estonia. Both systems employ cryptography and digital signatures to ensure the security and anonymity of the voting process. They also both allow voters to change their vote, with only the most recent vote being counted. In both cases, there is a processing stage prior to the counting stage, where invalidated ballots from re-voting are removed, and valid ballots are anonymized before counting. Additionally, both systems provide mechanisms for auditors to re-execute the steps of the processing stage to verify that the same results are achieved. In the previous chapter, we discussed the challenges associated with re-execution, and we believe that the integrity verification method proposed in this paper could also be applied to VoteAgain.

Table 1 provides a summary of the key features and coercion-resistance capabilities of the systems discussed.

In conclusion, this chapter has highlighted the critical role of the processing stage in ensuring both the integrity and coercion-resistance of end-to-end verifiable i-voting systems. While many systems, such as Helios and its variants, focus on verifiability and election transparency, they often lack mechanisms to counter over-the-shoulder coercion effectively. Coercion-resistant systems like Civitas and Selections offer promising solutions through features like re-voting and fake credentials, but they introduce additional complexity and usability challenges that must be carefully managed.

The Estonian i-voting system's use of re-voting as a coercion countermeasure, combined with its robust processing stage, positions it uniquely among other verifiable voting systems. The ability to re-vote, anonymize ballots, and provide verifiable processing steps ensures higher levels of voter privacy and election integrity. Similarly, the VoteAgain scheme, with its 2023 extensions, shares many of these features and demonstrates how scalable, coercion-resistant i-voting can be achieved.

Ultimately, this chapter demonstrates that while no system is perfect, Estonia's approach, along with systems like VoteAgain, represents a strong model for balancing verifiability, security, and resistance to coercion. The methods proposed in this article for verifying the processing stage could be applied across a range of systems to further strengthen election integrity.

VI. PROPOSING A METHODOLOGY FOR STANDARDIZING SETS FOR I-BALLOT-BOX INTEGRITY VERIFICATION

Because manual verification of i-ballot-boxes is impractical due to their size, and the re-execution of processing steps does not directly confirm the integrity of the i-ballot-box, an alternative method is necessary. In this section, we propose a method for verifying the integrity of the i-ballot-box. While auditors will need software due to the volume of data, unlike the processing software, this tool's sole purpose would be to verify integrity. Ideally, auditors could implement the integrity-checking software themselves. Still, it could also be an additional module within the existing open-source audit application previously used for verifying the results of the mixing and counting stages.

A. BRINGING BALLOTS TO COMPARABLE FORMS

There is no single unique identifier to reconcile ballots from input logs containing invalidated ballots and output logs containing anonymized ballots. However, all logs and sets of anonymized ballots have something unique to facilitate reconciliation. For our comparison, we had three unique values to choose from: the vote ID, the content of the ballot, or the checksum of the ballot obtained from its content using the sha256sum command. The vote ID was the first to be ruled out, as it is a reference value assigned to the ballot by the system, and lacks a strong correlation with the ballot's content itself. While comparing the content of ballots requires a byte-by-byte comparison, comparing checksums requires computing these sums once and then swiftly comparing them. In most logs, the checksum of the ballot is already logged instead of the ballot itself because it is unique, although the original ballot cannot be derived from it. Only in the log of improperly encrypted ballots is the checksum absent. We decided to use checksums instead of ballots for comparison because otherwise, finding the corresponding ballot for the checksum in the logs would essentially mean comparing based on checksums anyway. Based on the vote ID in the log of improperly encrypted ballots, we found the corresponding ballot in the input i-ballot-box and obtained its checksum. Since the log of improperly encrypted ballots has been empty so far or has had only a few entries, auditors can review these vote ID replacements one by one. As the correct use of the Voter Application should not result in improperly encrypted ballots, auditors should review all of these erroneous cases. There were two options for obtaining the checksum from the input i-ballot-box: either taking the ballot and finding the checksum with the sha256sum command or taking the checksum from the signature file. When a file is signed, only its checksum is signed and not the entire file. Obtaining the checksum in both ways should yield the same result, which can be independently verified. For anonymous ballots, the checksum can be found using the sha256sum command.

B. METHODS FOR FINDING CHECKSUMS OF BALLOTS

We need to find checksums for the four types of ballots to check i-ballot-box integrity. Below are the methods for each type along with the main steps.

1) CHECKSUMS OF ORIGINAL BALLOTS

The goal is to find checksums for all ballots in the i-ballot-box collected by the Collector Service after the voting period. Main steps:

- Load all files from the i-ballot-box zip file.
- Filter out only the .bdoc containers from the found files, which contain packed and signed files.
- Load only .ballot files from each .bdoc file, containing the vote hash in binary form.
- Compute the SHA-256 hash for the content of each .ballot file.
- Add the obtained hash to the set of returned checksums.

2) CHECKSUMS OF ANONYMOUS BALLOTS

The goal is to find checksums for the anonymous ballot box presented as a JSON-formatted file. Main steps:

- Load data from the ballot box JSON file.
- Traverse structured data at the ballot level, three levels below the districts.
- Decode each vote from base64 format to get the original binary representation.
- Compute the SHA-256 hash based on the decoded vote.
- Add the obtained hash to the set of returned checksums.

3) CHECKSUMS OF IMPROPERLY ENCRYPTED BALLOTS

The goal is to find checksums for improperly encrypted ballots using the error file in CSV format and the original i-ballot-box file in Zip64 format. Main steps:

- Open the error file using a CSV reader with tab ('\t') as a delimiter.
- Read the improperly encrypted vote data from the file.
- Load all files from the i-ballot-box zip file.
- Filter out only the .bdoc containers from the found files, whose name is an identifier of some erroneous vote.
- Load only .ballot files from each .bdoc file.
- Compute the SHA-256 hash for the content of each .ballot file.
- Add the obtained hash to the set of returned checksums.

4) CHECKSUMS OF LOGGED BALLOTS

The goal is to find checksums for logged ballots from a CSV-formatted log file. Main steps:

- Open the log file using a CSV reader with tab ('\t') as a delimiter.
- Analyze each row in the log file.
- If a row has fewer than five columns, move to the next row.
- Decode the second column of the row from base64 format to obtain the original binary representation.
- Add the obtained hash to the set of returned checksums.

C. METHOD FOR I-BALLOT-BOX INTEGRITY CHECKING

To check the integrity of the i-ballot-box, we first find checksums for all necessary ballots using the methods introduced in the previous section. We then show which combinations of sets of checksums must be equal for the i-ballot-box to be considered intact. All sets used in the following integrity checks, and how they are derived, are depicted in FIGURE 3.

Starting with finding checksum sets for ballots:

- Set of checksums for original ballot box – B_1: The result of the method for finding checksums of original ballots, using the i-ballot-box as the input for the processing stage.
- Set of checksums for anonymized ballot box – B_2: The result of the method for finding checksums of anonymized ballots, using the anonymized ballot box obtained by the end of the processing stage as input.

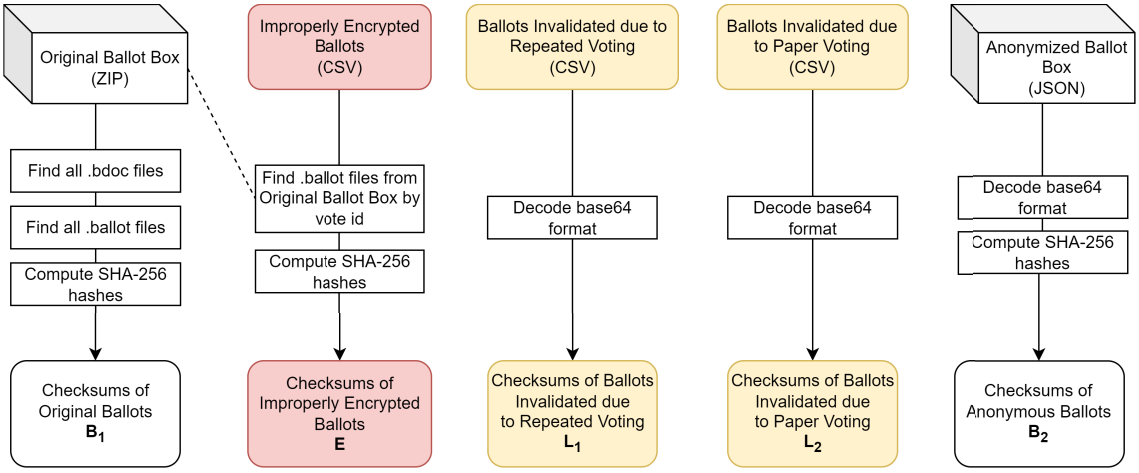

FIGURE 3. Transforming ballots for comparison.

- Set of checksums for improperly encrypted ballots – E: The result of the method for finding checksums of improperly encrypted ballots, using the i-ballot-box as input and the error file found in the squash step of the processing stage. If there is no error file, this set is empty.
- Set of checksums for ballots invalidated due to repeated voting – L_1: The result of the method for finding checksums of logged ballots, using ballots invalidated due to repeated voting logged in the squash step of the processing stage as input.
- Set of checksums for ballots invalidated due to paper voting – L_2: The result of the method for finding checksums of logged ballots, using ballots invalidated due to paper voting logged in the revoke step of the processing stage as input.

The equality formula for i-ballot-box integrity:

$$B_1 \setminus E = B_2 \cup L_1 \cup L_2 \,. \tag{1}$$

where the left side represents the input i-ballot-box ballots, excluding improperly encrypted ballots. The right side requires that all properly encrypted input ballots must be divided into anonymized ballots and ballots invalidated due to repeated voting or paper voting.

VII. VALIDATING THE INTEGRITY FORMULA FOR I-BALLOT-BOX

In this chapter, we explore the various scenarios in which the proposed integrity formula for the i-ballot-box was tested. Each scenario demonstrates specific conditions and results that help assess the robustness of the formula. Additionally, we suggest further checks to enhance the validation process.

Both the formula and all test scenarios are based on the assumption that the original ballot box, obtained as input for the processing stage, is neither empty nor manipulated; otherwise, there would be no basis for comparison. If ballots were manipulated during the voting stage, these manipulations would not be detected by the following test scenarios. At the end of the voting stage, the checksum of the i-ballot-box zip file is signed. During the processing stage, we trust the correctness of B_1 because it can be read directly from the original ballot box zip file after verifying that the checksum matches the signed checksum.

Since the formula compares sets, it assumes that all ballots in the original ballot box are unique and that the ballots in the groups resulting from the processing stage are also unique. If duplicates exist, this would either indicate an issue with the voting application or manipulation of the ballots. However, as duplicate checks were not implemented previously, we must account for the possibility that duplicates may exist within the ballot groups when considering the test scenarios.

A. THE ORIGINAL BALLOT BOX USED IN TEST SCENARIOS

In this section, we introduce the i-ballot box that was used as the input for the processing stage in all of the following test scenarios. The ballot box contains 10 unique ballots cast by four voters, representing all votes before any processing. It includes all the e-votes submitted by the voters during the election period.

The distribution of ballots among the voters was as follows:

- **Voter X**: 4 ballots (repeated votes).
- **Voter Y**: 2 ballots (one improperly encrypted and one correctly encrypted).
- **Voter Z**: 3 ballots (repeated votes).
- **Voter W**: 1 ballot (voted only once).

This setup reflects the typical scenario in an election system that allows for re-voting. Each voter can submit multiple votes, but only the last valid vote will be counted.

The initial list of ballots can be represented as follows:

$$\text{OriginalBB} = [x^1, x^2, x^3, x^4, y^1, y^2, z^1, z^2, z^3, w^1]\,.$$

Here, each superscript denotes a distinct vote submitted by the corresponding voter. For example, x^1, x^2, x^3, x^4 represent the four distinct ballots submitted by Voter X, which reflect repeated votes, while w^1 denotes the single vote cast by Voter W.

In the following test scenarios, this ballot box serves as the basis for testing the integrity of the i-ballot-box.

B. INTEGRITY FORMULA

The integrity formula, as defined in Formula 1, asserts that the original set of ballots (minus improperly encrypted ballots) should equal the union of anonymized valid ballots and invalidated ballots due to repeated or paper voting. To determine these sets, we first examine how the ballots were divided into groups after the processing stage:

- **Improperly encrypted ballots:** Only one ballot from Voter Y was improperly encrypted:
$$\text{ErrorB} = [y^1]\,.$$
- **Ballots invalidated due to repeated voting:** Repeated ballots from the same voter, excluding their most recent ballot. Ballot y^1 is excluded here because it was already classified as an improperly encrypted ballot:
$$\text{InvalidRepeated} = [x^1, x^2, x^3, z^1, z^2]\,.$$
- **Ballots invalidated due to paper voting:** Voter Z cast a paper ballot, invalidating their e-vote:
$$\text{InvalidPaper} = [z^3]\,.$$
- **Anonymized ballot box:** The final ballots from each voter, except for Voter Z, whose e-vote was invalidated due to paper voting:
$$\text{AnonymizedBB} = [x^4, y^2, w^1]\,.$$

We now compute the sets of checksums based on the division of ballots into these groups. The checksums for each group are represented as follows:

- Set of checksums for original ballot box (B_1):
$$B_1 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^1, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$
- Set of checksums for improperly encrypted ballots (E):
$$E = \{y_c^1\}\,.$$
- Set of checksums for ballots invalidated due to repeated voting (L_1):
$$L_1 = \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\}\,.$$
- Set of checksums for ballots invalidated due to paper voting (L_2):
$$L_2 = \{z_c^3\}\,.$$
- Set of checksums for anonymized ballot box (B_2):
$$B_2 = \{x_c^4, y_c^2, w_c^1\}\,.$$

The number of elements in each set corresponds to the number of ballots in the respective group, as all ballots in these groups were unique.

Next, we verify the validity of Formula 1.

Step 1: Calculate $B_1 \setminus E$:

$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

Step 2: Calculate $B_2 \cup L_1 \cup L_2$:

$$B_2 \cup L_1 \cup L_2 = \{x_c^4, y_c^2, w_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}\,.$$

Step 3: Simplify and reorder the sets for comparison:

$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

The sets match, so the formula holds.

C. TEST SCENARIOS

Below are the test scenarios used to validate Formula 1. Each case demonstrates specific conditions and results, helping to ensure the robustness of the formula. Both manipulated and non-manipulated cases are presented.

1) ALL GROUPS CONTAIN BALLOTS (NON-MANIPULATED)

This scenario acts as a baseline for validating the integrity formula in a non-manipulated case. Ensuring the formula holds in this situation establishes a foundation for detecting abnormalities in subsequent, manipulated scenarios.

Condition: Each group contains ballots as expected after the processing stage, with all groups populated.

Expected Outcome: In this non-manipulated scenario, the integrity formula holds without any discrepancies.

Verification:

- The calculations for $B_1 \setminus E$ and $B_2 \cup L_1 \cup L_2$ were demonstrated earlier in the chapter.
- Both sides of the formula are equal:
$$B_1 \setminus E = B_2 \cup L_1 \cup L_2\,.$$

This confirms that the formula works correctly in the expected, non-manipulated baseline case.

2) MOVING BALLOTS BETWEEN GROUPS (MANIPULATED)

This scenario demonstrates the behavior of the integrity formula when ballots are moved between logical groups. Specifically, it shows that while the formula can handle empty sets, it does not detect manipulations where ballots are moved between groups.

Condition: A ballot (z^3) is moved from InvalidPaper group to AnonymizedBB group, leaving InvalidPaper group empty. This manipulation results in the following sets:

$$L_2 = \{\}\,,$$
$$B_2 = \{x_c^4, y_c^2, z_c^3, w_c^1\}\,.$$

Expected Outcome: Even though a manipulation occurred, the integrity formula will still hold because it is designed to ensure no ballots were added, removed, or replaced during the processing stage—not to detect movement between groups. This scenario also demonstrates the formula's robustness in handling empty sets.

Verification:

- The initially calculated set $B_1 \setminus E$ remains unchanged:
$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$
- Recalculate $B_2 \cup L_1 \cup L_2$:
$$B_2 \cup L_1 \cup L_2 = \{x_c^4, y_c^2, z_c^3, w_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{\}.$$
- Simplify and reorder:
$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$
- Both sides of Formula 1 are equal, confirming:
$$B_1 \setminus E = B_2 \cup L_1 \cup L_2.$$

This case demonstrates that while the formula passes verification, it does not detect manipulations where ballots are moved between groups. Additional checks beyond the integrity formula are needed to identify such issues.

3) ADDING UNIQUE BALLOTS TO A GROUP (MANIPULATED)

This scenario demonstrates what happens when one or more unique ballots are added to a group, leading to changes in the corresponding set. The goal is to show that the integrity formula fails when additional ballots are improperly introduced during processing.

Condition: A new ballot (z^4) is added to the group AnonymizedBB, resulting in the following set B_2:
$$B_2 = \{x_c^4, y_c^2, w_c^1, z_c^4\}.$$

Expected Outcome: The integrity formula fails because the added ballot changes the contents of the set B_2, altering the relationship between the original set $B_1 \setminus E$ and the union of the resulting sets.

Verification:

1) The initially calculated set $B_1 \setminus E$ remains unchanged:
$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$
2) Recalculate $B_2 \cup L_1 \cup L_2$:
$$B_2 \cup L_1 \cup L_2 = \{x_c^4, y_c^2, w_c^1, z_c^4\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}.$$
3) Simplify and reorder:
$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, z_c^4, w_c^1\}.$$
4) Compare $B_1 \setminus E$ and $B_2 \cup L_1 \cup L_2$:
$$B_1 \setminus E \neq B_2 \cup L_1 \cup L_2.$$

The integrity formula fails because the added ballot z^4 changes the size and content of the set B_2, despite being logically added to the group AnonymizedBB.

4) REMOVING BALLOTS FROM A GROUP (MANIPULATED)

This scenario demonstrates what happens when a ballot is improperly removed from a group, leading to changes in the corresponding set. The goal is to show that the integrity formula fails when ballots are deleted during processing.

Condition: Ballot (x^4) from the AnonymizedBB group is removed. The updated set B_2 is:
$$B_2 = \{y_c^2, w_c^1\}.$$

Expected Outcome: The integrity formula fails because the removed ballot changes the contents of the set B_2, altering the relationship between the original set $B_1 \setminus E$ and the union of the resulting sets.

Verification:

1) The initially calculated set $B_1 \setminus E$ remains unchanged:
$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$
2) Recalculate $B_2 \cup L_1 \cup L_2$:
$$B_2 \cup L_1 \cup L_2 = \{y_c^2, w_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}.$$
3) Simplify and reorder:
$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$
4) Compare $B_1 \setminus E$ and $B_2 \cup L_1 \cup L_2$:
$$B_1 \setminus E \neq B_2 \cup L_1 \cup L_2.$$

The integrity formula fails because the removed ballot x^4 alters the size and content of the set B_2, despite being logically removed from the group AnonymizedBB.

5) REPLACING BALLOTS IN THE GROUPS WITH UNIQUE BALLOTS (MANIPULATED)

This scenario explores cases where one or multiple ballots in a group are replaced with unique ballots. It demonstrates that simple count-based validation fails to detect such replacements because the total number of ballots remains unchanged. However, checksum-based comparisons reliably catch such manipulations, as the replaced ballots alter the contents of the sets.

Condition: Ballot (w^1) in the AnonymizedBB group is replaced with a unique ballot (v^1), resulting in the following set B_2:
$$B_2 = \{x_c^4, y_c^2, v_c^1\}.$$

Expected Outcome: The integrity formula does not hold because the manipulation changes the set B_2, breaking the equality between $B_1 \setminus E$ and $B_2 \cup L_1 \cup L_2$.

Verification:

- The initially calculated set $B_1 \setminus E$ remains unchanged:
$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}.$$

- Recalculate $B_2 \cup L_1 \cup L_2$:

$$B_2 \cup L_1 \cup L_2 = \{x_c^4, y_c^2, v_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}\,.$$

- Simplify and reorder:

$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, v_c^1\}\,.$$

- Compare the two sets:

$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,,$$
$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, v_c^1\}\,.$$

  The two sets are not equal, confirming that the manipulation is detected by the integrity formula.

This case demonstrates that checksum-based validation is effective in detecting replacements of ballots in groups with unique ballots not previously present.

6) REPLACING BALLOTS IN THE GROUPS WITH DUPLICATE BALLOTS (MANIPULATED)

This scenario explores cases where one or multiple ballots in the AnonymizedBB group are replaced with duplicate ballots from either the invalid groups or within the AnonymizedBB group itself. It demonstrates that while simple count-based validation fails to detect such replacements, the integrity formula can reliably catch these manipulations.

Condition: Ballot (x^4) in the AnonymizedBB group is replaced with a ballot (x^2) from the InvalidRepeated group. This manipulation results in the following set B_2:

$$B_2 = \{x_c^2, y_c^2, w_c^1\}\,.$$

Expected Outcome: The integrity formula does not hold because the manipulation changes the set B_2, breaking the equality between $B_1 \setminus E$ and $B_2 \cup L_1 \cup L_2$.

Verification:

- The initially calculated set $B_1 \setminus E$ remains unchanged:

$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

- Recalculate $B_2 \cup L_1 \cup L_2$:

$$B_2 \cup L_1 \cup L_2 = \{x_c^2, y_c^2, w_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}\,.$$

- Simplify and reorder:

$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

- Compare the two sets:

$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,,$$
$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

  The two sets are not equal, confirming that the manipulation is detected by the integrity formula.

This case demonstrates that checksum-based validation is effective in detecting replacements of ballots in the AnonymizedBB group with duplicate ballots from either the invalid groups or within the AnonymizedBB group itself.

7) ADDING DUPLICATE BALLOTS TO THE VALID GROUP (MANIPULATED)

This scenario explores cases where one or multiple duplicate ballots are added to the AnonymizedBB group. It demonstrates that while the integrity formula alone fails to detect such additions due to the nature of sets (which store unique elements), combining it with a simple count-based validation can reliably catch these manipulations.

Condition: One or more duplicate ballots are added to the AnonymizedBB group. For instance, an additional x^4 is added to the AnonymizedBB group. This manipulation results in the following set B_2:

$$B_2 = \{x_c^4, y_c^2, w_c^1\}\,.$$

Expected Outcome: The integrity formula holds because sets store each element only once, so the duplicate ballot does not change the set. However, a simple count-based validation would detect the discrepancy in the number of ballots.

Verification:

- The initially calculated set $B_1 \setminus E$ remains unchanged:

$$B_1 \setminus E = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

- Recalculate $B_2 \cup L_1 \cup L_2$:

$$B_2 \cup L_1 \cup L_2 = \{x_c^4, y_c^2, w_c^1\} \cup \{x_c^1, x_c^2, x_c^3, z_c^1, z_c^2\} \cup \{z_c^3\}\,.$$

- Simplify and reorder:

$$B_2 \cup L_1 \cup L_2 = \{x_c^1, x_c^2, x_c^3, x_c^4, y_c^2, z_c^1, z_c^2, z_c^3, w_c^1\}\,.$$

- Compare the two sets:

$$B_1 \setminus E = B_2 \cup L_1 \cup L_2\,.$$

The integrity formula holds because sets store each element only once, so the duplicate ballot does not change the set. This case demonstrates that additional checks beyond the integrity formula are needed to identify such issues.

In the next section, we will discuss combining the integrity formula with count-based validation to enhance manipulation detection.

D. COMBINING INTEGRITY FORMULA WITH COUNT-BASED VALIDATION

To enhance the detection of manipulations, we additionally propose combining the integrity formula with a simple count-based validation. This combined approach can detect scenarios where duplicate ballots are added, which the integrity formula alone might miss.

Referring to the previous scenario where duplicate ballots were added to the AnonymizedBB group, we can see how count-based validation helps in detecting such manipulations.

Count-Based Validation:

- Verify that the count of the original ballot box (OriginalBB) matches the count of B_1:

$$|\text{OriginalBB}| = |B_1| = 10\,.$$

- Calculate the total number of ballots after processing:

$$|\text{AnonymizedBB}| = 4\,,$$
$$|\text{InvalidRepeated}| = 5\,,$$
$$|\text{InvalidPaper}| = 1\,,$$
$$|\text{ErrorB}| = 1\,,$$
$$4 + 5 + 1 + 1 = 11\,.$$

- Compare the counts:

$$10 \neq 11\,.$$

By combining the integrity formula with count-based validation, we can detect manipulations involving the addition of duplicate ballots, ensuring a more robust verification process.

These examples demonstrate how manipulated and non-manipulated cases are addressed, offering a comprehensive understanding of the robustness and limitations of Formula 1.

VIII. DISCUSSION

This chapter focuses on discussing and interpreting the research findings and their practical and theoretical implications. It also addresses the limitations of the study and presents conclusions and recommendations. These elements help to better understand the security issues related to Estonia's i-voting system and propose potential solutions.

A. SUMMARY OF FINDINGS

The key findings of the study include the identification of vulnerabilities in Estonia's i-voting system that could enable insider attacks and compromise the integrity of the i-voting box during the processing stage. The study proposes an auditing methodology to address these vulnerabilities and enhance the system's security. The importance of end-to-end verifiability and coercion resistance is emphasized, along with the need for continuous evaluation and improvement in the field of digital democracy. The findings answer the research questions by providing insights into the risks of insider attacks and the need for measures to ensure the integrity of the voting process. The proposed auditing methodology offers a framework for auditors to improve the security of the voting process, contributing to the reliability and transparency of internet voting systems. The study also highlights the limitations of existing coercion-resistant systems and underscores the need for further research and improvement in the field of e-voting. Overall, the findings contribute to understanding and improving the security of the i-voting system.

B. INTERPRETATION OF RESULTS

The results of the study have significant implications for the security and integrity of Estonia's i-voting system. The study identifies vulnerabilities that could be exploited by insider attackers during the processing stage, highlighting the need for robust security measures. These findings are consistent with previous studies that have also emphasized the importance of continuous assessment and improvement of system security. The study proposes an auditing methodology to enhance system security by verifying the integrity of the i-voting box. This theoretical contribution adds to the field of digital democracy by providing a practical solution. The study also underscores the importance of end-to-end verifiability and coercion resistance in the i-voting system, acknowledging the limitations of existing coercion-resistant systems. Overall, the study's results offer valuable insights, propose practical measures, and contribute to the theoretical understanding of digital democracy and i-voting security.

C. PRACTICAL AND THEORETICAL IMPLICATIONS

The practical and theoretical significance of the findings lies in their potential to improve the security and integrity of i-voting systems, not only in Estonia but also in other countries. By identifying vulnerabilities and proposing practical measures, the study contributes to the field of digital democracy and offers valuable insights for designing and evaluating secure and transparent voting systems. The proposed auditing methodology for verifying the integrity of the i-voting box can be adapted and applied to other internet voting systems, enhancing their security and reliability. Overall, the findings have practical implications for improving the trustworthiness of the voting process and theoretical implications for understanding the security of e-voting systems.

D. LIMITATIONS

While the study provides valuable insights into the security of Estonia's i-voting system, it is important to acknowledge its limitations and consider their impact on the reliability of the results.

Firstly, the study relies on the analysis of existing documents, protocols, and previous studies. The accuracy and completeness of these sources may affect the reliability of the findings.

Secondly, the use of sets in the proposed auditing methodology assumes that the checksums of all ballots are unique, which may not always be the case. While using a correct voter application should prevent this, custom voter applications or the duplication of existing ballots during the processing stage could introduce duplicates. Based on our test scenarios, we found that the proposed set-based auditing methodology may struggle specifically with detecting added duplicates. In such cases, we recommend combining the integrity formula with a simple count-based validation to enhance the detection of these manipulations. This combined approach allows auditors to identify discrepancies caused by added duplicates while maintaining the efficiency of the original set-based method.

Furthermore, the proposed auditing methodology does not guarantee that all ballots are correctly categorized as valid or invalid during the processing stage. This limitation means that the auditing methodology might not detect if a ballot is incorrectly categorized between valid and invalid

categories. Future work should include further checks on the categorization of valid and invalid ballots.

Given these limitations, it is important to interpret the study's results with caution and recognize the need for further research and validation to ensure the reliability and applicability of the proposed measures in real-world scenarios.

IX. CONCLUSION

This study conducted an evaluation of the security of Estonia's i-voting system, specifically focusing on its susceptibility to dishonest insider attacks. The primary objective was to scrutinize the audibility of the i-ballot-box integrity during the processing stage, where the i-ballot-box undergoes preparation for counting.

Through this evaluation, a potential vulnerability was identified, suggesting that a dishonest insider could manipulate election results by discreetly replacing ballots during the processing stage. To address this vulnerability, a solution was proposed, outlining a method for verifying the integrity of the i-ballot-box through the implementation of an additional auditing application.

It is essential to clarify that this study is framed within a theoretical context, exploring the hypothetical vulnerability of the i-voting system to certain manipulations. The intent is to contribute insights for enhancing the security of i-voting systems without assigning blame or imputing dishonest motives.

The validation of the proposed integrity verification formula across multiple scenarios proves its capability to detect a comprehensive range of manipulations. This work concentrated on verifying the integrity of the i-ballot-box, ensuring the preservation of all ballots, and preventing unauthorized additions at the conclusion of the processing stage.

Future investigations should extend to reviewing the mechanisms confirming the accurate categorization of valid and invalid ballots and also focus on applying the methodology to larger datasets and simulated election environments to confirm its scalability and robustness. Additionally, live pilot testing in controlled environments would provide valuable insights into its operational effectiveness and potential challenges in real-world applications.

ACKNOWLEDGMENT

For text editing and linguistic clarity, they utilized OpenAI ChatGPT[1]. This artificial intelligence system enabled them to swiftly obtain English-language text corrections and linguistic assistance.

REFERENCES

[1] Ü. Madise and T. Martens. (2006). *E-voting in Estonia 2005. The First Practice of Country-Wide Binding Internet Voting in the World*. Gesellschaft für Informatik e.V. [Online]. Available: https://dl.gi.de/handle/20.500.12116/29155

[2] P. Ehin, M. Solvak, J. Willemson, and P. Vinkel, "Internet voting in Estonia 2005–2019: Evidence from eleven elections," *Government Inf. Quart.*, vol. 39, no. 4, Oct. 2022, Art. no. 101718. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0740624X2200051X

[3] State Electoral Office Estonia. (2023). *Statistics About Internet Voting in Estonia*. [Online]. Available: https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia

[4] State Electoral Office Estonia. (2023). *IVXV Online Voting System*. [Online]. Available: https://github.com/valimised/ivxv

[5] State Electoral Office Estonia. (2023). *IVXV I-Voting Handbook*. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV%20e-h%C3%A4%C3%A4letamise%20k%C3%A4siraamatpdf

[6] State Electoral Office Estonia. (2023). *General Description of the Framework of the I-Voting System IVXV*. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV%20elektroonilise%20h%C3%A4%C3%A4letamise%20%C3%BCldraamistik.pdf

[7] State Electoral Office Estonia. (2023). *IVXV Protocols*. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV-protocols.pdf

[8] KPMG Baltics OÜ. (2023). *Elektroonilise Hääletamise Protsessi Auditeerimine*. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-05/L%C3%B5pparuanne%20RKV23%20EValimiste%20Audit.asice

[9] H. Lipmaa and O. Mürk. (2001). *E-valimiste Realiseerimisvõimaluste Analüüs*. [Online]. Available: https://www.valimised.ee/sites/default/files/uploads/eh/lipmaamyrk.pdf

[10] T. Tammet and H. Krosing. (2001). *E-Valimised Eesti Vabariigis: Võimaluste Analüüs*. [Online]. Available: https://www.valimised.ee/sites/default/files/uploads/eh/evalimisteanalyys24okt.doc

[11] T. Martens. (2003). *E-Hääletamise Organisatsiooniline Ja Tehniline Kontseptsioon*. [Online]. Available: https://www.valimised.ee/sites/default/files/uploads/eh/Kontsept-03.pdf

[12] J. Benaloh, R. Rivest, P. Y. A. Ryan, P. Stark, V. Teague, and P. Vora, "End-to-end verifiability," 2015, *arXiv:1504.03778*.

[13] R. L. Rivest, "On the notion of 'Software independence' in voting systems," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 366, no. 1881, pp. 3759–3767, Aug. 2008, doi: 10.1098/rsta.2008.0149.

[14] S. Heiberg, T. Martens, P. Vinkel, and J. Willemson, "Improving the verifiability of the Estonian internet voting scheme," in *Electronic Voting* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2017, pp. 92–107. [Online]. Available: https://www.researchgate.net/profile/Robert-Krimmer/

[15] B. Zhang, Z. Li, and J. Willemson, "UC modelling and security analysis of the Estonian IVXV internet voting system," 2021, *arXiv:2109.01994*.

[16] J. Willemson, "Bits or paper: Which should get to carry your vote?" *J. Inf. Secur. Appl.*, vol. 38, pp. 124–131, Feb. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212617304933

[17] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, "Security analysis of the Estonian internet voting system," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 703–715, doi: 10.1145/2660267.2660315.

[18] E-Valimiste Turvalisuse tööhm. (2019). *E-Valimiste Turvalisuse Tööhma Koondaruanne*. [Online]. Available: https://www.mkm.ee/en/media/8583/download

[19] ODIHR Election Expert Team. (2023). *Estonia, Parliamentary Elections, 5 March 2023: Final Report*. [Online]. Available: https://www.osce.org/files/f/documents/f/f/551179_0.pdf

[20] A. Sutopo, T. Haines, and P. Roenne, "On the auditability of the Estonian IVXV system," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2024, pp. 19–33. [Online]. Available: https://fc23.ifca.ai/voting/papers/SHR23.pdf

[21] T. Haines and P. Roenne, "New standards for E-voting systems: Reflections on source code examinations," in *Int. Conf. Financial Cryptogr. Data Secur.* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2021, pp. 279–289. [Online]. Available: https://eprint.iacr.org/2021/391.pdf

[22] M. Lindeman and P. B. Stark, "A gentle introduction to risk-limiting audits," *IEEE Secur. Privacy*, vol. 10, no. 5, pp. 42–49, Sep. 2012. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6175884

[1] OpenAI (2024) GPT-3.5. OpenAI ChatGPT https://www.openai.com/

[23] M. Bernhard, "Risk-limiting audits: A practical systematization of knowledge," in *Proc. 6th Int. Joint Conf. Electron. Voting*, 2021, pp. 162–177. [Online]. Available: https://www.researchgate.net/profile/David-Duenas-Cid/publication/355107346_Sixth_International_Joint_Conference_on_Electronic_Voting_E-Vote-ID_2021_-_Proceedings/
[24] M. Bernhard, J. Benaloh, J. Alex Halderman, R. L. Rivest, P. Y. A. Ryan, P. B. Stark, V. Teague, P. L. Vora, and D. S. Wallach, "Public evidence from secret ballots," 2017, *arXiv:1707.08619*.
[25] S. Popoveniuc, J. Kelsey, A. Regenscheid, and P. Vora, "Performance requirements for end-to-end verifiable elections," in *Proc. Electron. Voting Technol. Workshop/Workshop Trustworthy Elections*, Washington, DC, USA, Aug. 2010. [Online]. Available: https://www.usenix.org/conference/evtwote-10/performance-requirements-end-end-verifiable-elections
[26] B. Adida, "Helios: Web-based open-audit voting," in *Proc. 17th USENIX Secur. Symp.*, San Jose, CA, USA, 2008, pp. 335–348. [Online]. Available: https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf
[27] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proc. ACM Workshop Privacy Electron. Soc.*, New York, NY, USA, Nov. 2005, pp. 61–70, doi: 10.1145/1102199.1102213.
[28] D. Chang, A. K. Chauhan, M. K. Noufal, and J. Kang, "Apollo: End-to-end verifiable voting protocol using mixnet and hidden tweaks," in *Information Security and Cryptology—ICISC 2015*, S. Kwon and A. Yun, Eds., Cham, Switzerland: Springer, 2016, pp. 194–209, doi: 10.1007/978-3-319-30840-1_13.
[29] A. J. Perez and E. N. Ceesay, "Improving end-to-end verifiable voting systems with blockchain technologies," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1108–1115. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8726502
[30] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *Proc. IEEE Symp. Secur. Privacy (sp)*, May 2008, pp. 354–368. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4531164
[31] J. Clark and U. Hengartner, "Selections: Internet voting with over-the-shoulder coercion-resistance," in *Financial Cryptography and Data*, G. Danezis, Ed., Berlin, Germany: Springer, 2012, pp. 47–61. [Online]. Available: https://ifca.ai/pub/fc11/04.pdf
[32] A. Kiayias, T. Zacharias, and B. Zhang, "End-to-end verifiable elections in the standard model," in *Advances in Cryptology—EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds., Berlin, Germany: Springer, 2015, pp. 468–498. [Online]. Available: https://eprint.iacr.org/2015/346.pdf
[33] S. Bistarelli, I. Mercanti, P. Santancini, and F. Santini, "End-to-end voting with non-permissioned and permissioned ledgers," *J. Grid Comput.*, vol. 17, no. 1, pp. 97–118, Mar. 2019, doi: 10.1007/s10723-019-09478-y.
[34] S. Zhang, L. Wang, and H. Xiong, "Chaintegrity: Blockchain-enabled large-scale e-voting system with robustness and universal verifiability," *Int. J. Inf. Secur.*, vol. 19, no. 3, pp. 323–341, Jun. 2020, doi: 10.1007/s10207-019-00465-8.
[35] X. Boyen, T. Haines, and J. Müller, "Epoque: Practical end-to-end verifiable post-quantum-secure E-voting," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Sep. 2021, pp. 272–291. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9581185
[36] N. M. Shakiba, M.-A. Doostari, and M. Mohammadpourfard, "ESIV: An end-to-end secure internet voting system," *Electron. Commerce Res.*, vol. 17, no. 3, pp. 463–494, Sep. 2017, doi: 10.1007/s10660-016-9230-y.
[37] M. Kumar, S. Chand, and C. P. Katti, "A secure end-to-end verifiable internet-voting system using identity-based blind signature," *IEEE Syst. J.*, vol. 14, no. 2, pp. 2032–2041, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8977363
[38] P. Y. A. Ryan, P. B. Rønne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D.Wallach, M. Brenner, Eds., Berlin, Germany: Springer, 2016, pp. 176–192, doi: 10.1007/978-3-662-53357-4_12.
[39] M. B. Verwer, I. Dionysiou, and H. Gjermundrød, "TrustedEVoting (TeV) a secure, anonymous and verifiable blockchain-based e-voting framework," in *E-Democracy-Safeguarding Democracy and Human Rights in the Digital Age*. Cham, Switzerland: Springer, 2020, pp. 129–143, doi: 10.1007/978-3-030-37545-4_9.
[40] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "VoteAgain: A scalable coercion-resistant voting system," in *Proc. 29th USENIX Secur. Symp.*, Aug. 2020, pp. 1553–1570. [Online]. Available: https://www.usenix.org/conference/
[41] T. Haines, J. Müller, and I. Querejeta-Azurmendi, "Scalable coercion-resistant E-voting under weaker trust assumptions," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, New York, NY, USA, Jun. 2023, pp. 1576–1584, doi: 10.1145/3555776.3578730.

TARVO TREIER was born in Võru, Estonia, in 1982. He received the B.Sc. and M.Sc. (cum laude) degrees in computer science from Tallinn University of Technology, Estonia, in 2005 and 2007, respectively, where he is currently pursuing the Ph.D. degree with the Department of Software Science.

Since 2013, he has been a Lecturer with the Department of Software Science, Tallinn University of Technology. He has also worked in several companies as a Software Engineer. With over 20 years of experience in software engineering, he specializes in streamlining workflows and enhancing efficiency. His research interests include e-government, distributed systems, and distributed applications.

KRISTJAN DÜÜNA was born in Rapla, Estonia, in 1994. He received the B.Sc. and M.Sc. (cum laude) degrees in computer science from Tallinn University of Technology, Estonia, in 2020 and 2024, respectively.

He has over ten years of experience in software engineering, involving distributed systems and integration projects. His research interests include e-government, cybersecurity, and cryptography.

Mr. Düüna achieved a top 100 ranking in the IEEEXtreme Programming Competition, in 2019 and 2020.

Appendix 2

II

T. Treier. Beyond the happy path: A safety-critical audit methodology for Estonia's i-voting processing application. *IEEE Access*, 13:203429–203443, 2025

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 November 2025, accepted 26 November 2025, date of publication 28 November 2025,
date of current version 5 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3638663

RESEARCH ARTICLE

Beyond the Happy Path: A Safety-Critical Audit Methodology for Estonia's I-Voting Processing Application

TARVO TREIER
Department of Software Science, Tallinn University of Technology, 12618 Tallinn, Estonia
e-mail: tarvo.treier@taltech.ee

ABSTRACT Auditing an Internet voting (i-voting) system can be viewed as a specialised form of software testing: the auditor acts as the tester, the goal is legal assurance rather than product release, and success depends on providing independently verifiable evidence. While Estonia's legally binding i-voting platform includes procedural checks, it currently lacks a systematic, test-driven audit methodology—particularly for the vote-processing stage, where integrity checks and anonymisation are critical. This study presents a structured, evidence-based audit methodology for that stage. It identifies 37 potential fault scenarios—from duplicate votes to silent manipulations—maps them to 19 functional requirements, and highlights where targeted tests are missing. A requirement–risk matrix is used to reveal these coverage gaps. To enable reproducible audits, the paper introduces a GDPR-compliant synthetic dataset: one reference ballot box and a set of fault-seeded variants designed to expose critical risks, including a manipulation undetected by the previously used integrity-check tool. The examples illustrate how such faults can be detected automatically and how auditors could use measurable coverage metrics to track improvements over time. The methodology is designed to allow qualified observers to replicate the same tests on the synthetic data without compromising ballot secrecy. The requirements list, fault catalogue, and dataset guidelines have been shared with-on an independent basis-the National Election Service for possible consideration in preparation for the 2025 local elections. These results indicate that safety-critical testing principles can be adapted to strengthen the practical auditability of the i-voting processing stage.

INDEX TERMS Auditability, end-to-end verifiability, Estonian i-voting, fault seeding, independent verification and validation, mutation testing, safety-critical systems, vote processing.

I. INTRODUCTION

Estonia has been a global pioneer in Internet voting (i-voting), using it in nationwide elections since 2005 [1], making election integrity and independent verification increasingly critical to public trust. The system's security model combines cryptographic protocols, strict procedural controls, and a carefully maintained balance between transparency and ballot secrecy. Over the years, the platform has evolved, adding individual and end-to-end (E2E) verifiability features [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

However, academic studies and international observation reports continue to highlight weaknesses in practical *auditability*—especially during the vote-processing stage, where invalid ballots are removed, encrypted votes are anonymised, and ballots are prepared for cryptographic mixing before the final tally [3], [4], [5]. Although the process should in theory yield sufficient evidence for independent replay, documented audits reveal gaps in fully verifying non-ideal—so-called ''unhappy-path''—scenarios.

Although i-voting is not life-critical in the narrow technical sense, its *societal* criticality is comparable: a silent failure can delegitimize elected authority, destabilize institutions, and erode public trust for years. In aviation and healthcare,

rare but high-impact failures drive assurance practices that privilege evidence, independence, and measurable coverage over informal confidence. Elections face analogous risks—low-probability adversarial faults with high consequence—so adopting discipline from safety-critical assurance provides a structured way to ground correctness claims in independently checkable evidence. This framing motivates the next subsection, which views electoral auditing explicitly as structured software testing grounded in traceable evidence.

On 18 February 2024, the author submitted a memorandum to the National Election Service listing 37 potential fault scenarios, recommending a purpose-built synthetic dataset, and proposing that every contesting party be allowed to appoint its own auditor under conditions equivalent to those of the official audit team. A refined requirements list and detailed dataset description followed on 12 June 2024.

The OSCE–ODIHR legal opinion of 17 June 2025 [6] later encouraged the authorities to carry out independent post-election audits and adopt transparent testing procedures for Estonia's system. Several of the measures proposed by the author in 2024 align with these recommendations, particularly the calls for independent post-election audits and broader stakeholder involvement.

A. AUDITING AS SOFTWARE TESTING

Auditing can be viewed as a specialised form of software testing. The auditor occupies the tester's seat, the goal is legal approval rather than product release, and success means producing evidence that qualified observers can independently scrutinise—at least on a synthetic dataset identical to that used by auditors. Framed this way, proven high-integrity testing techniques become directly relevant, yet they are still largely absent from current practice, which currently relies heavily on a dedicated integrity-check tool, among other procedural and technical checks. Because auditors have had no access to fault-seeded input data, they cannot confirm how well this tool detects realistic error cases—an uncertainty illustrated by test case 26-2, which the tool fails to flag.

B. GAPS IN CURRENT PRACTICE

A review of public audit reports, source code, and test-election procedures confirms that current practices do not systematically cover the 37 fault scenarios documented by the author through code and documentation analysis, which range from duplicate or missing ballots to manipulated logs and compromised ballot-box integrity. Existing procedural checks detect some nominal faults, yet no structured, evidence-based testing framework addresses the full range of critical error cases. The methodology introduced here addresses this gap by explicitly covering, in its initial phase, 17 high-priority scenarios drawn from the full set of 37 documented cases, presented as exemplars to demonstrate the method rather than as a comprehensive catalogue.

C. CONTRIBUTION OF THIS PAPER

This paper addresses the outlined gaps through three main contributions:

1) a *requirement–risk matrix* explicitly linking fault scenarios to functional requirements and highlighting coverage gaps;
2) an *independent audit workflow* combining DO-178C structural coverage and mutation-testing principles to define measurable auditing goals;
3) clear, *reproducible guidelines* for creating a GDPR-compliant synthetic dataset, enabling independent test execution and comparable coverage metrics.

The fault catalogue (18 February 2024) and subsequent detailed requirements list and dataset guidelines (12 June 2024) have been shared-on an independent basis-with the National Election Service for possible consideration in preparation for the 2025 local elections. Its practical application in this paper (Section VI) illustrates Phase A of the methodology; subsequent phases are outlined for future work. The remainder of this paper presents the regulatory context (Section II), a verifiability analysis (Section III), related work (Section IV), detailed methodology (Section V), its Phase A application (Section VI), and future directions (Section VII).

II. THE FRAMEWORK OF ESTONIAN I-VOTING

According to the general framework for i-voting in Estonia [7], every eligible voter can cast a ballot over the Internet using an ID card or another recognised electronic identity (eID) method that supports authentication and digital signing. The legal framework requires secrecy, equality, and security at least equivalent to traditional paper voting. Voters may cast multiple electronic votes; only the *last* one is counted, and any subsequent paper ballot overrides all electronic votes (e-votes). A separate mobile application allows voters to verify ballot delivery and encryption. While the overall architecture is outlined here for context, the audit methodology developed in this paper is applied specifically to the processing stage, where the documented fault scenarios primarily arise.

A. STAGES OF ESTONIAN I-VOTING

The process is divided into four organisational stages [7]:

1) **Pre-voting**: system preparation and generation of election-specific encryption and decryption keys.
2) **Voting**: voters cast e-votes; in-person voting may run in parallel.
3) **Processing**: integrity and authenticity of the Internet ballot box (i-BB) are verified, duplicates and paper-overridden e-votes are removed, and the remaining ballots are anonymised and cryptographically mixed.
4) **Counting**: the mixed and anonymised ballots are decrypted and tallied.

B. SYSTEM COMPONENTS

The main components involved in these stages are summarised below [7].

- **Voter application**: encrypts, signs, and sends ballots to the vote-collection service.
- **Verification application**: lets the voter confirm that the encrypted ballot stored by the system matches the intended choice.
- **Vote-collection service**: stores ballots in the i-BB and obtains trusted timestamps from the registration service. After voting closes, the resulting i-BB is cryptographically signed and the data transferred to processing.
- **Registration service**: issues the timestamps that link each ballot to a reception time.
- **Processing application** (focus of this study): verifies i-BB integrity, invalidates duplicates and paper-overridden ballots, compiles voter lists, and anonymises the votes. This is the stage where the majority of the 37 documented fault scenarios identified by the author could manifest.
- **Mixing application**: shuffles anonymised votes so that they can be counted publicly without revealing voter choices.
- **Key-management application**: generates the election keys and decrypts the anonymised votes for the final tally.

C. THE PROCESSING APPLICATION IN DETAIL

Vote processing is performed in an offline processing environment [7]. All input data (i-BB, logs and configuration) arrive on removable media. All outputs, including the anonymised vote set and voter lists, are written back to the same medium. Before any output leaves the offline environment, a cryptographic hash value (message digest) is generated and digitally signed on an Internet-connected workstation, then returned to the offline processing environment. Key processing steps are:

1) Verify the signed hash of the i-BB and validate the digital signature of each ballot contained within it.
2) Cross-check timestamps from the registration service with collection-service logs.
3) Remove duplicate e-votes, retaining only the latest one per voter.
4) Invalidate e-votes if the voter later cast a paper ballot.
5) Anonymise ballots by separating signatures from encrypted choices.
6) Produce voter lists and anonymised ballots for transfer to the mixing application.

D. WHY THE PROCESSING APPLICATION NEEDS STRONGER EVIDENCE

The election service publishes several audit tools on GitHub [8], including a log-based integrity-check tool that grew out of earlier research [9]. Although valuable, auditors are not supplied with a fault-seeded test corpus to independently verify the tool's sensitivity to realistic error conditions. While auditors are free to create their own tests, doing so requires significant specialised effort, and the official test elections focus primarily on nominal "happy-path" scenarios. In the author's experiment (test scenario 26-2), a deliberately crafted manipulation passed the integrity check without raising an alert, illustrating that the tool's effectiveness against certain non-nominal cases cannot be assumed without targeted testing. This uncertainty, combined with the lack of documented processing stage replays after real elections [10], motivates the fault-seeded, evidence-based audit approach developed in the rest of the paper.

III. VERIFIABILITY ANALYSIS

The verifiability and security of Estonia's i-voting system have been central topics in multiple academic studies and technical assessments. This section reviews the current state of verifiability, its relationship to end-to-end (E2E) requirements, the mechanisms available for auditing the processing application, and the practical constraints imposed by privacy and coercion-resistance guarantees.

A. DEFINITION AND IMPORTANCE OF E2E VERIFIABILITY

In 2015, the U.S. VOTE Foundation's report on secure i-voting [11] recommended that all public elections conducted online must be E2E verifiable. E2E verifiability is a *property*; one family of i-voting schemes aiming to realise it in practice is often referred to as *E2E-VIV*. Such schemes let voters confirm that their ballots were cast as intended and recorded as cast, while enabling universal verification that all recorded ballots were included in the final tally as recorded. Most E2E-VIV systems are explicitly designed to eliminate the need for public trust in the correctness of server-side operations under the control of election officials or their delegates [11].

A key requirement for such systems is *software independence*: the correctness of the election result must not depend on the proper functioning of any software, and E2E-VIV schemes are therefore engineered to be software-independent [11]. As Rivest defines it, a software-independent voting system is one in which any software error or malicious change cannot alter the election outcome without leaving detectable evidence [12]. This concerns the basis of trust, not the use of tools: audits may employ software, but correctness must rest on independently checkable artefacts and evidence. In contrast, a software-dependent system remains vulnerable to silent and undetectable manipulation—especially during complex processing stages where data are transformed and filtered. Preventing such silent theft is a central objective of E2E verifiability.

Estonia's IVXV platform, introduced in 2017, is presented as E2E-verifiable. Heiberg et al. claimed that the scheme ensures both individual verification and overall transparency [2]. A later study, however, concluded that the

system achieves only partial E2E verifiability, balancing it against coercion resistance [13].

Even when an E2E system generates ample evidence for detecting problems, this alone is not sufficient. As Bernhard et al. emphasize [14], the evidence must actually be examined to verify correctness. Estonia's IVXV platform does produce a detailed audit trail, yet this trail requires active and independent scrutiny to be effective. Clarke similarly argues that E2E mechanisms may fail if implementations are flawed or if audits are not conducted independently on behalf of all stakeholders [15]. Therefore, E2E verifiability must be assessed not only in theory but also in practice—based on whether such scrutiny is feasible and whether it is actually being performed with sufficient rigour and independence.

B. CURRENT VERIFIABILITY STATUS OF ESTONIA'S I-VOTING SYSTEM

Estonia allows each voter to verify an e-vote within 30 minutes of casting by using a QR code that queries the vote-collection service. The ballot's integrity is protected by the voter's digital signature. Timestamps from the registration service help detect vote loss or alteration, provided that the collection and registration services do not collude [2]. The scheme also relies on the assumption that voters will perform post-casting verification themselves.

Despite these measures, full E2E verifiability is not achieved [13]. The system remains partially dependent on server-side processes and their auditability. The remaining limitations primarily stem from an inherent tension—discussed in Subsection III-D—between coercion resistance and full public auditability, a trade-off that imposes structural limits on how transparently the entire process can be verified in practice.

C. VERIFICATION PROCEDURES AND AUDITING MECHANISMS

IVXV introduced a dedicated auditor role responsible for confirming that every intended ballot is included in the i-BB and correctly handled. After voting ends, the collection service hands the i-BB to the processing application. Auditors must verify both the integrity of this input and the correctness of all subsequent steps.

The source code of the processing application is publicly available on GitHub [8]. Auditors can conduct code reviews and monitor test elections run by the election service, in which the choices of all test voters are predetermined and publicly known. Heiberg et al. further noted that the processing application is *intended to be re-executed* after the election to confirm reproducibility [2], yet no public audit report documents any such re-run [10], [16].

Auditors may perform random spot checks to verify whether selected ballots were valid, contained all required security elements, and were placed in the correct category. The election service also publishes audit tools for the mixing and key-management applications, as well as the log-based integrity-check tool. In practice, however, these tools are typically executed only within the service's managed environment, limiting opportunities for fully independent validation. Although the code is publicly available and can be rebuilt off-site, meaningful external testing has so far been hindered by the lack of fault-seeded datasets—a gap that the test corpus introduced in Section VI is specifically designed to address.

D. LIMITATIONS IN VERIFYING THE PROCESSING APPLICATION

Certain features—particularly re-voting—introduce fundamental constraints. Because a later paper ballot overrides an e-vote, each electronic ballot must be linked to a personal identifier until duplicates are resolved. Publishing such data would breach voter anonymity and violate statutory secrecy [17]. Re-voting mitigates over-the-shoulder coercion [18], but it does not prevent attack patterns such as Last-Minute Voter Coercion [19], which exploit the final moments before polls close. Even so, the option to re-vote lowers coercion risk by giving voters a practical means to override a coerced ballot with one that reflects their true preference. An auditing mechanism must therefore expose errors without revealing identities or the number of times each voter cast an e-vote.

The election service's integrity-check tool addresses part of the problem, yet auditors currently lack access to a fault-seeded test corpus to independently verify the tool's sensitivity to realistic error conditions. Creating realistic test cases requires specialised expertise and often exceeds the practical time constraints of official audits. In the author's experiment (scenario 26-2), a deliberately crafted manipulation passed the integrity check without raising an alert, illustrating that the tool may not detect certain non-nominal cases without targeted testing. Many of the 37 documented fault scenarios identified by the author concern the processing stage, where ballots are filtered, anonymised, and prepared for mixing. This result, combined with the lack of documented replays of the processing application after real elections, reinforces ODIHR's 2025 call for independent post-election audits [6]. Whenever a trade-off arises between voter privacy and fraud detection, the U.S. VOTE Foundation advises resolving it in favour of privacy [11].

Consequently, the following sections propose a GDPR-compliant, fault-seeded, evidence-based audit workflow that respects voter anonymity while enabling reproducible and rigorous verification of the processing stage.

IV. OVERVIEW OF PREVIOUS WORK

The auditability of Estonia's i-voting processing application has attracted sustained attention in academic studies, observation reports, and security analyses. Building on Section III, which examined general verifiability challenges, this section reviews prior work that focuses specifically on auditing the processing stage. The literature highlights

recurring limitations such as documentation focused primarily on ''happy-path'' scenarios, restricted auditor access, and methodological gaps that constrain fully independent verification.

A. ISSUES IN AUDITING THE PROCESSING APPLICATION

Springall et al. showed in their 2014 security study that early Estonian control procedures assumed fault-free execution and provided no specific guidance for handling failure cases [3]. Juvonen's 2019 comparative analysis reached a similar conclusion, noting that official documentation described nominal paths in detail but offered little coverage of off-nominal scenarios—commonly contrasted with the ''happy path'' in software engineering, which refers to a typical, error-free run with no unexpected inputs [4], [20].

International observers have echoed these concerns. The OSCE/ODIHR 2023 report found that the critical step of removing e-votes overridden by re-voting or paper ballots had never been audited [5]. ODIHR's 2025 legal opinion reiterated the need for independent post-election audits and transparent testing modalities [6]. Similarly, the 2024 cybersecurity risk assessment by the Estonian Academy of Sciences listed the potential lack of auditing or observation during certain i-voting stages as a medium-level risk. One low-level risk involved a scenario in which an auditor executes an unauthenticated version of the processing application—or fails to verify its authenticity. This could, under certain conditions, allow the application to alter or delete votes [21].

B. ROLE AND LIMITATIONS OF AUDITORS

The verifiability of the processing application depends heavily on the access rights and trustworthiness of designated auditors. Barański et al. emphasise that individual, universal, and eligibility verifiability all rely on auditors and on the level of trust placed in them [22]. Müller argues that threat scenarios—particularly those concerning privacy, verifiability, and coercion resistance—remain insufficiently defined in both academic literature and public documentation, raising concerns about the transparency and rigour of the system's audit model as a whole [23].

Boone notes that the Estonian election service mandates that comprehensive audits be conducted within an organiser-controlled environment. This setup restricts external reproducibility and undermines universal verifiability, as no independently generated proof of correctness is made available [24]. Furthermore, the current audit process may not reliably detect whether ballots are incorrectly retained or removed during processing. If such conditions were exploited, the integrity of the final tally might no longer be independently verifiable by third parties.

C. TECHNICAL AND METHODOLOGICAL CHALLENGES

Sutopo et al. point out that, although IVXV is described as software-independent, it still relies on software to verify evidence and assumes that back-end attacks are out of scope [25]. They also highlight the presence of substantial unused or redundant code in the repository, which complicates audit review. Finogina et al. and Krips et al. emphasise that verifiability hinges on the assumption that at least some auditors are honest and competent [26], [27]. Treier et al. demonstrate that verifying the integrity of the ballot box after processing is difficult without additional tooling; their proposed log-based integrity check can detect added, removed, or replaced votes but does not cover all critical scenarios [9].

This study addresses the methodological gap by supplying a GDPR-compliant, fault-seeded corpus and a requirement–risk matrix, enabling independent tool evaluation on published artefacts without reliance on organiser-controlled environments. The approach is architecture-agnostic—no changes to back-end databases, hardware, or system design are required—and it neither assumes nor excludes ledger-based components (e.g., blockchains).

Recent surveys organise end-to-end election evidence into mechanism classes spanning casting, recording, and tallying [28]. Complementary design-time work applies safety–security–privacy threat modelling to electronic voting architectures (STPA with STRIDE and LINDDUN) [29]. The present study aligns with the taxonomy's *Recording* class through set- and cross-artefact checks (O1, O6) and complements these frameworks by providing a public, reusable corpus with explicit oracles that focuses on evidence available today.

D. INTERNATIONAL PARALLEL: SWITZERLAND

In 2019, independent reviewers demonstrated critical verification flaws in the Scytl/Swiss Post system that could have allowed a server-side insider to read or alter votes [30]. As a consequence, the platform was not used in the May and October 2019 federal votes [31]. After suspension, the system returned under stronger public scrutiny, including full source publication, a public verifier, and multi-round third-party re-examinations operating on published specifications, code, and artefacts conducted by the Bern University of Applied Sciences (BFH) [32]. The same reports note that the available information did not yet suffice for third parties to build a fully independent verifier and that the verification chain was not evidenced end to end [32].

Swiss practice already includes executed tests and an explicit catalogue of verifier checks maintained with the code [32]. A sample of the developer's documented functional tests is executed by examiners to validate results [32]. The catalogue of implemented verification steps has grown across releases, but BFH observed identifier inconsistencies and recommended moving the catalogue into the verifier specification and formalising the checks [32].

A comparable silent-manipulation class of risk was identified for Estonia and later published by Treier et al. [9]. An integrity-check tool was deployed prior to public disclosure to mitigate set-consistency anomalies [9].

Coverage remained incomplete for the processing stage, and in a fault-seeded snapshot used in this article one implementation-level variant (F26–2) passed without a flag, indicating that targeted stimuli are required to expose certain cases (see Section VI).

Complementing the Swiss catalogue-and-testing practice, the use of a public and reusable corpus of *fault-seeded artefacts* exercises reject paths and borderline cases against published tools, with machine-checkable expected outcomes. Checks are expressed as explicit *oracles* that bind inputs to accept or reject decisions, enabling independent sensitivity evaluation outside organiser-controlled environments. Where expert controls and test materials are available, they can be packaged as fault-seeded catalogues, enabling reuse across cycles and making coverage reviewable rather than assumptive. Taken together, these elements add reproducibility, independent re-use, and measurable coverage to the existing verifier-centric approach (see Sections V and VI). Viewed this way, a proactive, evidence-driven posture reduces the risk that Estonia faces a Switzerland-style pause after a silent-manipulation issue has been demonstrated.

V. AUDITING THE PROCESSING APPLICATION AS SOFTWARE TESTING

Several previous sections have shown that the reliability of Estonia's i-voting depends heavily on auditors. The auditor's role is to confirm that the processing application—i.e., handover and validation of the i-BB, duplicate and paper-override handling, and preparation for mixing—operated in accordance with requirements and documentation. The Riigikogu Election Act requires the State Electoral Office to organise an audit in which an information-systems auditor reviews the functioning of the test election and the correctness/integrity of vote processing [17]. At the same time, academic literature and observation reports have stressed that current auditing does not cover the entire process and is not fully independent (see Subsections IV-A and IV-B).

Given that the auditor assesses the correspondence between inputs and outputs, auditing the processing application can be treated as a specialised form of software testing. Practices from safety-critical domains show that rigorous testing combines multiple phases and approaches [33]. Under DO-178C-style assurance, these map to concrete verification activities (e.g., requirements-based testing in normal and robustness modes, structural coverage, data/control-coupling analysis, and appropriate independence) that can be adapted to the i-voting context [34], [35], [36]. In this setting, correctness is grounded in externally verifiable artefacts rather than on trust in the correctness of any single software component. This perspective provides structure and concrete techniques to mitigate risks and increase confidence where errors would have serious consequences.

A. PARALLELS WITH ACCEPTANCE, FACTORY, AND SITE TESTING

Estonia's test elections resemble acceptance testing [37], where a system is tested against functional requirements. These exercises are typically closed-box tests that focus on observable input–output behaviour [38]. Because test elections are performed by the auditor in an organiser-controlled environment, they correspond closely to factory acceptance testing [33] and align with the notion of Alpha testing [39]. In certain safety-critical domains, these activities are complemented by Beta-style exercises in which independent end-user representatives execute tests outside the developer's environment [33], [39].

In the i-voting setting, the auditor formally acts as the end-user's representative. Although auditors have had source-code access and carried out sample checks in test and live elections, no public record shows comprehensive, systematic testing in an *independent* environment that validates their tools' ability to detect a representative set of faults from the documented fault catalogue.

B. REQUIREMENTS AND PRACTICES FOR TESTING CRITICAL SOFTWARE

The i-voting system is best considered *mission-critical* in Fowler's sense—failures can undermine the successful completion of the electoral process and institutional trust [40]. Where manipulation of outcomes could plausibly pose risks to people's lives, i-voting can reasonably be treated as *safety-critical* [41]; applying safety-critical assurance principles is therefore a conservative but justified choice. The U.S. VOTE Foundation likewise recommends following engineering practices established for mission- and safety-critical systems when designing E2E-verifiable schemes [11].

Safety-critical domains rely on layered methods: unit, integration, system and regression testing, stress testing, automated and manual checks, and code review [33]. This constitutes a holistic testing culture that goes beyond the "happy path" and explicitly targets fault domains, exceptions, and adversarial behaviours.

NASA-STD-8739.8B defines Independent Verification and Validation (IV&V) as rigorous analysis and testing to produce objective evidence, carried out with technical, managerial, and financial independence [35]. In the electoral setting, this independence breaks the circular-trust problem in which the organiser controls both the environment and the tools used to verify it. IV&V explicitly considers both nominal and off-nominal (non-standard) conditions to provide reliable input on system fitness [35].

DO-178C—the international aviation software standard—emphasises requirements-based testing, independence, and measurement against coverage *criteria*, together with disciplined fault seeding for hazard analysis [34], [42]. Effective verification under DO-178C requires each test to be linked to a specific requirement, include coverage accounting, and be performed independently [34]. Moy et al. explain that DO-333—the Formal Methods Supplement to DO-178C—enables formal, evidence-based reasoning by replacing structural coverage metrics with four goal-driven activities: coverage, completeness, data-flow control, and detection of extraneous code [36].

In the i-voting context, where auditing is often data-centric (logs, i-BB snapshots, signatures) rather than based on live execution, these goals translate into:

1) **requirement–to–test traceability and coverage accounting** (*coverage*);
2) **a comprehensive catalogue of off-nominal scenarios with corresponding tests** (*completeness*);
3) **end-to-end traceability from inputs to outputs across the processing stage** (*data-flow control*); and
4) **recognition that detecting unreachable or unjustified code typically requires source-code analysis beyond a data-only audit** (*extraneous code*).

Mutation-testing principles can be adapted to assess audit sensitivity in the i-voting context. Unlike traditional mutation testing, which alters the system's source code (''code mutants''), the proposed approach leaves the processing software untouched and uses pre-constructed, fault-seeded input–output scenarios with explicit oracles. Concretely, a *mutant* is a fault-seeded variant of an otherwise valid artefact (e.g., an i-BB snapshot or log) constructed by applying a small, targeted transformation that models a documented fault class (F_i). Examples include permuting registration timestamps to induce misordering, swapping ballot records, or altering a signature container while preserving format consistency. As Offutt describes [43], mutation testing strengthens a test set by introducing deliberate anomalies and then checking whether existing procedures detect them (i.e., whether the mutant is ''killed''). In the audit setting, a mutant is *killed* when the specified oracle raises the expected anomaly or classification (e.g., O_3 flags an inclusion/consistency violation); equivalent mutants that cannot affect oracle outcomes are excluded from scoring. To reduce the risk of unintentionally equivalent cases, artefact mutants should be associated with at least one oracle that is expected to detect them; in practice, this can be enforced, for example, by manually reviewing hand-constructed mutants against the oracle set. Scenarios can be prepared by independent experts, the election service, or the research community and shared as a public test corpus; independence is preserved because evaluation runs on identical artefacts with explicit oracles.

By operating entirely on published artefacts and explicit oracles, this closed-box adaptation allows auditors to validate their own tools without relying on internal application logic or organiser-controlled execution environments—directly addressing the independence gap noted in Section IV. Seeded scenarios (fault seeding/mutation) are mapped to requirements and checkpoints, outcomes are judged against expected behaviour for the specified inputs (an explicit oracle in specification-based testing) [38], and when formal methods are used, DO-333 provides alternative activities to achieve coverage goals [36]. Progress can be tracked using scale-independent measures: percentage of requirements with at least one associated test (R-coverage), percentage of fault classes detected per oracle (F-coverage), and the mutation score for the exercised corpus.

C. A STRUCTURED AUDIT METHODOLOGY ADAPTED FROM MISSION- AND SAFETY-CRITICAL PRINCIPLES

Building on the observations above and the practices described in Subsection V-B, this article proposes a structured and evidence-based audit methodology for Estonia's i-voting processing application. The methodology aims to increase audit rigour, scope, transparency, and reliability, moving beyond happy-path checks and producing verifiable evidence of both correct behaviour and fault response. It adapts principles from DO-178C [34], NASA IV&V [35], and safety-critical best practices [33], [42], [43], and consists of three interrelated activities.

1) PREPARATION: REQUIREMENT MAPPING, RISK ANALYSIS, AND TEST DESIGN

Systematically identify functional and security requirements for the processing application from system documentation, legal texts, and source code (when available). Associate each requirement with concrete checkpoints and logic branches, and ensure that every test is traceable to a specific requirement in line with adapted DO-178C principles. Compile a requirement–risk matrix and a catalogue of potential faults, anomalies, and adversarial scenarios (off-nominal paths), providing guidance for their identification. Design test cases that cover both compliant behaviour and identified risks, with particular emphasis on fault seeding to probe hazardous conditions [42]. Apply mutation-testing principles by constructing input–output ''mutants'' with explicit oracles that simulate small but critical deviations, and plan to record whether audit procedures detect their effects (i.e., whether the mutant is killed) [43]. Classify each fault class as *demonstrated by exemplar*, *implied by an invariant* (e.g., checksum equality), or *deferred to later phases*, and define acceptance thresholds for critical classes.

2) EXECUTION: INDEPENDENT TESTING AND TOOL VALIDATION

Provide auditors with datasets and documentation sufficiently early to establish an *independent* test environment. Validate both the processing application (or relevant components) and the auditor's own tools and procedures, analogous to Site or Beta testing [33], [39]. Because these exercises run on published artefacts and explicit oracles, validation of audit tools can proceed without privileged access to or control over the application's internal mechanisms. Use automated tooling and targeted manual checks to assess integrity (e.g., presence of required security elements, cryptographic compliance, log consistency). Track progress with coverage metrics against both requirements and the documented fault catalogue, prioritising critical logic branches and reporting against DO-178C coverage *criteria* [34]. Report R-coverage, per oracle F-coverage, and mutation score for the exercised corpus; where applicable, include per-oracle failure-condition logs to support analysis and robustness improvements. Ensure technical, managerial, and financial independence as required

by NASA-STD-8739.8B, and involve auditors with diverse expertise to improve robustness and credibility [35].

3) VALIDATION AND REPORTING: EVIDENCE-BASED TRANSPARENCY

Support post-election replay and re-execution of the processing application using the same official inputs, with previously validated tools and methods, to demonstrate reproducibility under audit constraints. Document all phases, test cases, results, anomalies, and evidence comprehensively, with clear statements of scope, methodology, findings, and limitations. Publish non-confidential artefacts—such as methodology descriptions, test narratives, coverage metrics, and anonymised findings—so that qualified observers can scrutinise the process without compromising secrecy or personal data. Where feasible, provide oracle failure-condition summaries (why a check did not trigger) to aid reproducibility and independent challenge. In short, assurance must be grounded in an open, reproducible evidence trail—comprising replayable inputs, documented test narratives, and measurable coverage metrics—rather than relying solely on technical fixes without procedural hardening.

The proposed audit workflow is divided into three phases—Preparation, Execution, and Validation & Reporting—each with clearly defined activities and measurable outputs. Figure 1 illustrates the sequence and interdependencies of these phases, highlighting the progression from initial requirement mapping to the public disclosure of audit results.

VI. PHASE A: REQUIREMENTS, FAULT SCENARIOS, AND COVERAGE PLANNING

This section instantiates Phase A of the audit workflow by consolidating functional requirements for the processing application, compiling a catalogue of fault scenarios, mapping requirements to risks, introducing explicit detection *oracles*, and defining a synthetic test corpus. The artefacts are: a baseline list of nineteen requirements (R1–R19) drawn from official IVXV documentation, a catalogue of thirty-seven fault scenarios (F1–F37), a requirement–risk matrix, a set of oracles (O1–O6) that formalise detection conditions, and a GDPR-neutral corpus consisting of one reference i-BB and fault-seeded variants. These artefacts instantiate Phase A of the methodology proposed in Subsection V-C and demonstrate how test preparation can be performed without live system access, relying solely on artefact analysis. Phase B (Execution) and Phase C (Validation and Reporting) are intentionally left to future work. Phase A deliberately concentrates most of the specialist effort (deriving requirements, defining fault classes, constructing fault-seeded artefacts, and writing explicit oracles) into a reusable corpus. The goal is that subsequent audits reuse this corpus and its oracles rather than having to rediscover faults and handcraft stimuli from scratch for each election cycle, thereby constraining both time cost and expert labour cost during execution. Phase A complements mechanism catalogues and design-time threat modelling by providing dataset-level stimuli and explicit oracles that enable outcome decisions from published artefacts—independent of organiser-controlled execution environments—and remains architecture-agnostic, requiring no modification to the deployed system [28], [29].

Threat model. The Phase A corpus and its oracles target three risk classes: accidental processing defects, malicious insider manipulation, and silent tampering with ballot sets or metadata between steps. This explicitly includes media/format and transfer errors in artefacts (e.g., archive–hash mismatches, missing or swapped subfiles, signature/container profile issues, cross-artefact inconsistencies between i-BB, lists, logs, and registration bundles). The objective is not to prove such events impossible, but to require that any deviation leaves independently detectable evidence in published artefacts. Correctness is therefore grounded in reproducible, externally checkable data rather than trust in the organiser's execution environment or internal storage.

A. COMPILING THE REQUIREMENTS LIST

Following Phase A, formal functional requirements for the processing application were identified from official IVXV system documents [7], [44], [45], [46], [47]. The resulting set (R1–R19), grouped by theme, is as follows.

1) INITIALIZATION AND INPUT VERIFICATION

- R1 Use a valid, signed election configuration file. [44]
- R2 Ensure input readability and integrity pre-checks before processing. [46]

2) INTEGRITY CHECKS AND DATA CONSISTENCY

- R3 Validate each e-vote's digital signature. [7], [46]
- R4 Verify presence of a registration-service timestamp for each vote. [7]
- R5 Check consistency between collection and registration services. [45]
- R6 Detect and report inconsistencies between those services. [45]
- R7 Verify timestamps and certificate validity confirmations against the specified protocol. [47]

3) VOTER IDENTITY AND SIGNATURE CONTAINER CHECKS

- R8 Verify the voter appears on the valid list. [45]
- R9 Validate the BDOC container profile and structure. [47]
- R10 Verify the correct naming of the encrypted ballot within the container. [47]

4) VOTE SELECTION IDENTIFICATION AND CLEANSING

- R11 Identify re-votes and retain only the latest e-vote [7], [46]
- R12 Maintain the linkage between voter identity and the last e-vote. [7]
- R13 Invalidate e-votes for voters who also cast a paper ballot. [7]

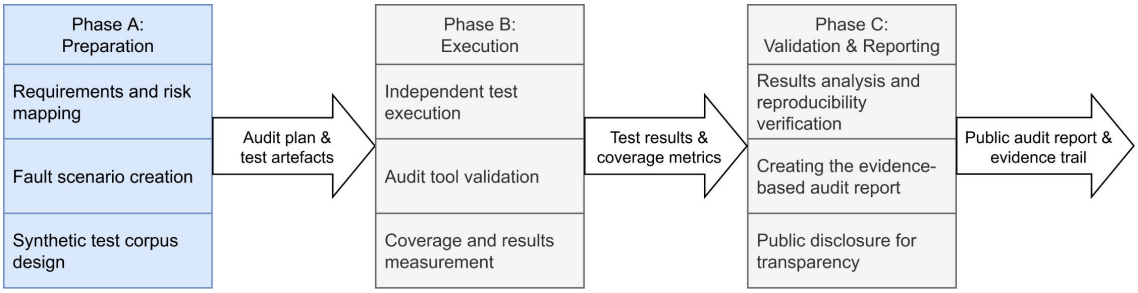

FIGURE 1. A three-phase, Evidence-based audit methodology.

R14 Ensure anonymisation occurs only after cleansing has completed. [7]

5) ANONYMISATION AND OUTPUT

R15 Anonymise e-votes while preserving necessary district attributes. [7], [45]
R16 Separate cryptograms from signatures as part of anonymisation. [45]
R17 Output anonymised e-votes in the correct, documented format. [45]

6) MANAGEMENT OF LISTS AND LOGS

R18 Generate both human-readable (PDF) and machine-readable lists of e-voters. [45]
R19 Produce activity and error logs. [45]

B. FAULT SCENARIO CATALOGUE

Identifying fault scenarios is part of test-case preparation, where each scenario serves as input for fault seeding or mutation testing. Based on a codebase review (140+ files at the time of analysis), documentation study, and prior work [9], thirty-seven fault scenarios (F1–F37) were identified and grouped as follows.

1) BALLOT-BOX INPUT ERRORS (F1–F17)

These concern i-BB contents and metadata at ingest. Examples: a signed i-BB ZIP archive hash that does not match the actual archive (F1); missing per-ballot subfiles such as `version`, `tspreg`, `ocsp`, or `bdoc` (F2); subfiles swapped between ballots (F3); oversized or invalid BDOC (F4, F7); missing or empty cryptograms (F5, F6); malformed cryptogram inside the BDOC (F16); BDOC profile mismatch (F17). Timestamp and identity anomalies include conflicts between BDOC signature time and registration-service timestamp (F8), confirmations signed with the wrong registration-service certificate (F9), e-votes cast outside the legal window—before start (F10) or after end (F11)—multiple identical cryptograms with the same confirmation (F12) or with different confirmations (F13), mismatches between the registration list and i-BB contents (F14), and e-votes from ineligible voters (F15).

2) ERRORS IN OTHER INPUT LISTS (F18–F24)

These involve voter lists, district definitions, cancellation lists, and registration bundles. Issues include missing or invalid signatures (F18, F20, F22), invalid file structure or contents (F19, F21, F23), and a signed ZIP archive hash for the registration bundle that does not match the actual archive (F24).

3) OUTPUT-RELATED SCENARIOS PRIOR TO MIXING (F25–F37)

These concern processing stage outputs before mixing. Integrity violations (adapted from [9]): incorrect electoral-district assignment (F25), addition of a new unique e-vote not present in the i-BB (F26), insertion of a duplicate (F27), removal of an existing e-vote (F28), replacement by a new unique e-vote (F29), replacement by a duplicate of another e-vote (F30). Validity manipulations: swapping an overridden (non-latest) e-vote's cryptogram with another voter's latest cryptogram from the same district (F31) or different district (F32), altering the order (F33) or timestamps (F34) of a multi-voter's e-votes, wrongful invalidation of a valid e-vote (F35), wrongful validation of a non-latest e-vote (F36), and failure to invalidate a voter's latest e-vote despite a paper vote (F37).

C. ORACLES AND COVERAGE MAPPING

In mutation testing and fault seeding, an *oracle* is a condition or rule that determines whether a given test case reveals a fault. Here, oracles are expressed as deterministic predicates over the artefacts (i-BB, lists, logs, outputs) that indicate whether a requirement violation has occurred. Using explicit oracles ensures that detection claims are verifiable without relying on subjective judgment.

The six primary oracles used in this methodology are:

O1 **Set consistency:** no unexpected additions, removals, or duplicates in defined sets (e.g., i-BB entries vs. registration list).
O2 **Last-vote invariant:** only the latest e-vote per voter is valid, unless invalidated by a paper vote.
O3 **Paper override:** all e-votes of a paper voter are invalidated.

TABLE 1. Minimal illustrative subset of the requirement–fault mapping (based on the current corpus and the author's mapping).

Pattern	Example	Status
One-to-one	R13 → F37 (paper override)	mapped
One-to-one	R10 → F5 (ballot file naming)	mapped
One-to-many	R11 → F31, F32, F33, F34, F35, F36 (last-vote rule)	mapped
Many-to-one	(R2, R3, R4) → F2 (input subfile presence/placement)	mapped
Req w/o fault	R6 → — (report cross-service inconsistencies)	gap
Fault w/o req	F13 → — (identical cryptogram with different confirmations)	gap

O4 **District mapping:** anonymised outputs preserve correct electoral-district attributes.

O5 **Signature and packaging validity:** all signatures and container structures match their specifications.

O6 **Cross-artefact consistency:** values in related artefacts (e.g., registration bundle vs. i-BB) match exactly.

1) FORMALISATION SKETCH

Each oracle is defined as a concrete, machine-checkable rule over published artefacts, not an informal judgement call by an auditor. For example, Oracle O2 (last-vote invariant) can be stated in operational terms as:

1) For every voter who also appears on the paper-vote list, none of that voter's electronic ballots may remain marked as valid in the final processed output.
2) For every other voter, exactly one electronic ballot may remain marked as valid, and it must be the one recorded as the latest for that voter.

A mutant in classes F33–F37 is considered "killed" by O2 if either of these conditions is violated in the artefacts given to the auditor (for example, an older ballot is still marked valid, or a paper voter still has an active electronic ballot). This makes the check objectively evaluable from the data alone.

The other oracles (O1, O3–O6) follow the same pattern: each is phrased as a deterministic condition on artefacts.

2) MAPPING APPROACH

A requirement–fault mapping (R1–R19 × F1–F37) was constructed to support requirement-to-test traceability and to identify coverage gaps. To keep the focus on what matters operationally, this section first introduces a *minimal illustrative subset* of mapping patterns in Table 1, and then lists the *current gaps* (requirements without a corpus stimulus; and fault classes without a mapped requirement) in Table 2. The mapping is a planning instrument for Phase B/C test design; it documents what the *corpus* exercises today and where additional stimuli are planned, rather than judging operational coverage of the deployed system.

3) COVERAGE SNAPSHOT (PHASE A)

R-coverage (requirements with ≥1 mapped fault class): **12/19 (63%)**. This percentage reflects the current design scope of the Phase A exemplar corpus and the author's initial fault selection; it *must not* be read as a statement about the IVXV system or existing audits. Table 2 summarises the gaps using short codes: "R-gap" = requirement without a corpus stimulus; "F-gap" = fault class without a mapped requirement.

TABLE 2. Summary of gaps in the requirement–fault mapping.

Code	ID	Short note
R-gap	R1	valid, signed election configuration (no stimulus yet)
R-gap	R6	detect/report cross-service inconsistencies (no stimulus yet)
R-gap	R14	anonymisation only after cleansing (no stimulus yet)
R-gap	R16	separation of cryptograms from signatures (no stimulus yet)
R-gap	R17	output format/schema conformance (no stimulus yet)
R-gap	R18	human-readable vs. machine-readable voter lists (no stimulus yet)
R-gap	R19	activity/error logs (no stimulus yet)
F-gap	F4	oversized/invalid BDOC
F-gap	F10	e-vote cast before start of the legal window
F-gap	F11	e-vote cast after end of the legal window
F-gap	F13	identical cryptogram with different confirmations
F-gap	F25	wrong electoral-district assignment in output
F-gap	F26	new unique ballot added (not present in i-BB)
F-gap	F27	duplicate ballot inserted
F-gap	F28	valid ballot removed
F-gap	F29	replaced by a new unique ballot
F-gap	F30	replaced by a duplicate of another ballot

4) SENSITIVITY SNAPSHOT (PHASE A EXEMPLAR CORPUS)

Using the publicly released log-based integrity-check tool, the exemplar corpus yielded a mutation score of **5/17**. Within its intended scope—*set-consistency* anomalies (**O1**)—the checker killed **5/6** exercised variants; the sole O1 miss was **F26-2**. For O2–O6 the score was **0**, as these anomaly types lie outside the tool's design focus. Accordingly, a non-trigger does *not* imply a requirement failure; it typically indicates an out-of-scope oracle for this tool. In a controlled reconstruction, re-executing the *processing application* would be expected to detect most exercised classes; time-window violations (e.g., F11) are not enforced by the processing application and would remain outside that mechanism's scope.

5) INTERPRETATION

A mapped link indicates a *candidate stimulus* present in the Phase A corpus; it does not by itself imply complete coverage of the requirement. One stimulus rarely suffices: thorough coverage typically requires multiple fault classes and multiple oracles. Importantly, these figures characterise the *exemplar corpus*, not the operational IVXV system or official audit practice. Some F-gaps (e.g., F13) indicate a

specification/documentation gap rather than an operational claim: the corpus models a plausible replay/uniqueness anomaly, but no explicit public requirement states an anti-replay or cryptogram-uniqueness rule. For the instantiated Phase A corpus, the target is **100%** detection of exercised fault variants (i.e., a 100% mutation score) and, as the corpus expands, **complete** R-coverage; large-scale confirmation and final stopping rules are Phase B/C deliverables. Phase B/C will extend the corpus to close the listed gaps and will report *R-coverage*, *per-oracle F-coverage*, and *mutation score* on executed tests.

D. TEST CORPUS: BASELINE DESIGN AND ARTEFACT MODEL

To enable reproducible and independent audit exercises, Phase A produces a synthetic test corpus. It follows two principles: (*i*) it is GDPR-neutral, using only publicly documented eID test identities [48]; and (*ii*) it is *artefact-only*, meaning all verification runs on static data files (i-BB, lists, logs) without requiring live execution of the processing application. The acceptance criterion is tooling-agnostic: re-executing the processing application in an independent environment, using a separate verifier, or performing assisted manual checks are all acceptable provided they consistently reproduce the stated oracle outcomes on the public fault-seeded corpus. For clarity, the dataflow and the minimal per-variant record are illustrated in Figure 2; inputs feed the unit under test (UUT), while the dataset's reference output is reserved for oracle checks, and the tool-level decision follows an any-reject rule.

1) USING THE FAULT-SEEDED CORPUS

Each dataset consists of *inputs* and a *reference output bundle* used only for oracle checks. The UUT is run solely on the inputs; oracle checks compare the UUT outcome to the reference bundle. For *Baseline*, the expected tool-level outcome is ACCEPT. For *fault-seeded* datasets, at least one oracle must REJECT; the tool-level decision is REJECT if any oracle rejects, otherwise ACCEPT.

2) PER-VARIANT RECORD

Each run yields exactly one minimal log row ⟨`dataset_id`, `unit_id`, `rejecting_oracle`, `decision`⟩. Here, `dataset_id` uniquely identifies the baseline or seeded set; `unit_id` identifies the audited implementation—e.g., a binary/image hash or Git commit for software, or a manual-protocol identifier with worksheet hash for manual checks; `rejecting_oracle` is empty on Baseline or lists one or more oracle identifiers on seeded sets; and `decision` is ACCEPT/REJECT. Aggregating these rows over runs produces the coverage and sensitivity summaries reported later.

3) BASELINE SCENARIO

A baseline scenario is defined to mirror valid processing inputs without manipulations. Seven eligible test voters are

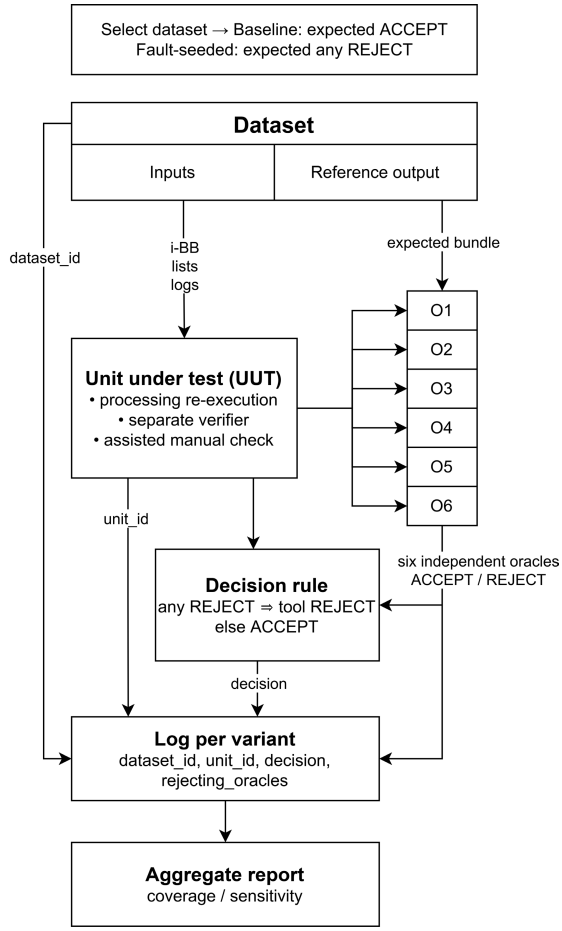

FIGURE 2. Execution workflow with the public corpus (multi-oracle).

TABLE 3. Test voters in Baseline (public Mobile-ID test identities).

Voter	Test ID	Dist.	#votes	Notes
X	60001017716	0001.1	4	Frequent re-voter
Y	60001017727	0001.1	2	Regular voter
Z	60001017738	0002.1	3	Paper vote override
W	60001017749	0001.1	1	Single submission
A	60001017750	0001.1	0	Eligible, no vote
B	60001017761	0002.1	0	Eligible, no vote
C	60001017772	0001.1	0	Eligible, no vote

defined, as detailed in Table 3; four participate, casting a total of ten e-votes. The scenario exercises re-voting and paper overrides while remaining small enough for manual verification.

4) EXPECTED CORRECT PROCESSING

Valid anonymised ballots: *three* (X last, Y last, W only). Invalidated by re-voting: *four* (X first three; Y first).

Invalidated by paper vote: *three* (all Z ballots). Baseline includes the i-BB, the registration bundle, voter/district/cancellation lists, and the expected outputs and logs from correct processing. This baseline serves as the ground truth from which all fault-seeded variants are derived.

E. CONSTRUCTING FAULT-SEEDED VARIANTS: PATTERNS AND RECIPES

Fault scenarios are instantiated as *mutants*: controlled edits to Baseline artefacts. Each variant has an explicit detection oracle (O1–O6, as defined in Subsection VI-C) and is linked to one or more functional requirements (R1–R19) for coverage tracking. Integrity violations (F25–F30) are adapted from prior work [9]. For most variants, a short description is sufficient to reconstruct the modification logic; however, a few (e.g., F26-2, F33, F34, F35) require additional explanation because they involve coordinated changes across multiple artefacts, ordering manipulations, or subtle data alterations. For each variant, record: ID and short name; targeted fault class(es) F_i; related requirement(s) R_j; primary oracle(s) O_k; construction pattern (A–C below); artefacts touched; expected oracle outcome; and killed/not-killed result. All edits are performed in a sandbox build under version control; exact diffs and build metadata are preserved. Modified binaries are not distributed.

1) PATTERN A — INVALID INPUT WITH BYPASSED CHECK (PRE-PROCESSING STIMULUS)

When to use: input-level faults (e.g., BDOC/profile/signature anomalies, cross-service inconsistencies, ineligible voter, out-of-window casting).

Steps: (1) Hand-edit the input artefact to violate the targeted constraint; (2) locate in the processing application the enforcement point for this check and temporarily disable or invert it; (3) run processing with the bypassed check on the modified input to obtain ''apparently correct'' outputs despite the faulty input; (4) save inputs, outputs, logs, and metadata.
Oracles: typically O5 (signature/container validity), O6 (cross-artefact equality), sometimes O1 for set-level effects.
Used in this corpus: F7, F14, F15.

2) PATTERN B — OUTPUT-ONLY MANIPULATION (POST-PROCESSING STIMULUS)

When to use: set membership, ordering, attribute-preservation, or classification anomalies that can be expressed in outputs/logs (e.g., add/remove/replace/duplicate, cross-district swaps, timestamp/order edits).

Steps: (1) Produce correct outputs from the Baseline; (2) hand-edit anonymised outputs and/or logs to introduce the targeted anomaly; (3) keep file structures and schemas consistent (JSON/CSV); (4) record metadata.

Oracles: O1 (set consistency), O2–O3 (last-vote and paper-override invariants), O4 (district/attribute preservation).

Used in this corpus: F25–F34.

3) PATTERN C — INSTRUMENTED PROCESSING FOR CLASSIFICATION ERRORS

When to use: misclassification cases that are hard to emulate purely in outputs (e.g., wrongful invalidation, non-latest retained, paper override not applied).

Steps: (1) Identify the decision point (e.g., `squash`/`revoke`) in code; (2) temporarily invert/bypass the decision; (3) run processing on the Baseline; (4) preserve emitted logs and record exactly what was toggled.
Oracles: O2 (last-vote invariant), O3 (paper override).
Used in this corpus: F35–F37.

4) CATALOGUE OF INSTANTIATED VARIANTS

Table 4 lists the Phase A mutants, showing for each the manipulation applied, the processing step affected, the ballots involved, and the *primary* oracle expected to detect the anomaly.

F. EXTENDED DESCRIPTIONS FOR SELECTED VARIANTS

For some fault-seeded corpus variants in Table 4, the compact descriptions do not fully convey the manipulations applied. Below are extended notes for selected cases.

1) F26-2: UNIQUE BALLOT ADDITION WITH LOG EDITS (BUILDS ON F26)

This variant builds directly on F26: a new unique ballot has already been added to the anonymised output (the set has changed). In addition, audit logs and aggregate counters are edited so that published hash totals appear consistent with the altered state, without introducing further per-ballot changes beyond F26. This probes whether audits bind logs and aggregates strictly to the actual output set, rather than accepting aggregate alignment alone.

Expected oracle: **O1** (primary: set consistency); *O6* (secondary: cross-artefact consistency).

2) F33: RE-ORDERING OF A REPEATED VOTER'S BALLOTS

For a re-voter (X), the third and fourth ballots are swapped in order in both the check-step logs and the i-BB JSON. Such reordering can influence last-vote determination if the procedure assumes chronological order without independently validating timestamps.

Expected oracle: **O2** (last-vote invariant).

3) F34: ALTERATION OF CASTING TIMESTAMPS FOR A REPEATED VOTER

Vote timestamps are altered so that an earlier ballot appears later and a later ballot appears earlier. For example, voter Y's first ballot is moved to a later time than the second, and voter X's last ballot is set earlier than the penultimate. This tests whether the audit process cross-validates timestamps or relies solely on them for determining the last valid ballot.

Expected oracle: **O2** (last-vote invariant).

TABLE 4. Selected fault-seeded corpus variants (compact catalogue).

ID	Manipulation	Step	Affected ballot(s)	Oracle
F7	Invalid BDOC signature	check	X–V4	O5
F14	Registration bundle vs. i-BB inconsistency	check	X–V4; Y–V2	O6
F15	Ineligible voter ballot	check	N–V1	O6
F25	Wrong electoral district	anonymise	B–V1	O4
F26	New unique ballot added	check/anonymise	Y–V3; V–V1	O1
F26-2	Unique addition + log edit†	check+logs	as F26	O1
F27	Duplicate ballot inserted	anonymise	X–V4P	O1
F28	Valid ballot removed	check/anonymise	Y–V2; X–V4	O1
F29	Replaced by new unique ballot	anonymise	W–V1 ← V–V1	O1
F30	Replaced by duplicate of other ballot	anonymise	X–V4 ← X–V2	O1
F31	Cryptogram swap (same district)	check	Y–V2 ↔ X–V1	O6
F32	Cryptogram swap (cross-district)	check	Y–V2 ↔ A–V1	O4
F33	Re-ordered votes for re-voter†	check	X–V3 after X–V4	O2
F34	Altered casting timestamps†	check	Y–V1 later than Y–V2	O2
F35	Valid ballot marked invalid†	squash/revoke	X–V4; Y–V2	O2
F36	Non-latest ballot retained	squash	X–V3	O2
F37	Paper override not applied	revoke	Z–V3	O3

† Extended description in Subsection VI-F.

4) F35: VALID BALLOT MARKED INVALID

Two valid ballots are incorrectly marked invalid—one during the `squash` step and one during the `revoke` step. The affected entries are added to the invalid-ballots list and removed from the subsequent i-BB version. This simulates wrongful invalidation that could alter the final tally but remain undetected if the audit compares only final states rather than reconstructing the full invalidation history.

Expected oracle: **O2** (last-vote invariant) or **O3** (paper override), depending on context.

VII. DISCUSSION

This study examined the auditability of Estonia's i-voting processing application and framed the processing-stage audit as a software-testing and verification task governed by mission- and safety-critical principles (Section V). A structured three-phase workflow was defined to organise preparation, independent execution, and evidence-based validation.

A. IMPLEMENTED SCOPE AND FINDINGS

For the analysed processing-application version, Phase A artefacts were instantiated: the requirements list (R1–R19), the fault-scenario catalogue (F1–F37), the requirement–risk mapping, six explicit oracles (O1–O6), and a GDPR-neutral, artefact-only test corpus. Seventeen representative test cases were executed as exemplars, covering a subset of requirement–fault links and illustrating oracle-based evaluation on public artefacts. The exercise revealed traceable requirement–risk links, highlighted coverage gaps, and exposed tool-scope limitations that become visible only under fault-seeded stimuli. Test-based auditing proved feasible in this domain when centred on artefacts rather than on organiser-controlled environments or purely functional tests.

B. ACHIEVED CONTRIBUTIONS

Safety-critical testing concepts were adapted to the processing-stage audit, producing an artefact-centred workflow rather than an environment-centred one. A traceable structure linking requirements (R1–R19), fault scenarios (F1–F37), and explicit oracles (O1–O6) was produced to support coverage planning and reporting. A GDPR-neutral public corpus was assembled that enables independent reproduction of accept/reject outcomes on published inputs and reference outputs and supports sensitivity summaries via mutation-style snapshots. Although instantiated for Estonia, the structure is transferable to other high-assurance contexts by substituting system-specific artefacts.

C. LIMITATIONS

Only 17 of the 37 documented fault scenarios were instantiated as test cases. Requirement and fault lists reflect the analysed code and documentation snapshot and require periodic updates. No new audit tools were implemented; the focus remained on artefacts and evaluation criteria. Practical guarantees for auditor independence were assumed rather than designed. A full formalisation of the artefact model and detection oracles, together with machine-checked correctness proofs, is left for future work.

D. FUTURE WORK

Extend Phase A to cover all requirement–fault links and enlarge the public corpus accordingly. Carry out Phase B by executing the corpus in an independent environment and validating tools against the seeded manipulations. Perform Phase C by analysing coverage, re-running processing post-election for reproducibility, and producing evidence-based reports. Develop open-source audit utilities guided by the mapped requirements and oracles. Where log-based integrity checks are already deployed, the fault-seeded corpus can be used either in assisted manual exercises or via

simple wrapper scripts around the existing checker, allowing the checker to be benchmarked on the corpus rather than replaced. Study governance and independence safeguards for operational settings. Update requirements, scenarios, and mappings after each system release to sustain currency.

VIII. CONCLUSION

Auditing of the i-voting processing stage can be operationalised as a structured software-testing task grounded in mission- and safety-critical principles. The defined three-phase, artefact-centric workflow and its Phase A instantiation—requirements, fault scenarios, explicit oracles, and a GDPR-neutral corpus—enable independent, reproducible, and evidence-based evaluation. Exemplar execution showed that fault-seeded stimuli and oracle checks reveal integrity and classification anomalies that conventional functional testing may miss. Sensitivity can be summarised via mutation-style snapshots, making coverage and misses auditable on public artefacts. The approach is tool-agnostic and compatible with independent re-execution, separate verifiers, or assisted manual procedures. Advancing Phases B and C will enable rigorous, transparent, and independently verifiable audits that reinforce trust in published election results.

ACKNOWLEDGMENT

The author used OpenAI's ChatGPT[1] to improve language and structure. All research ideas, analysis and conclusions are solely the author's.

REFERENCES

[1] P. Vinkel. (Apr. 2012). *Internet Voting: Experiences From Five Elections in Estonia*. [Online]. Available: https://www.researchgate.net/publication/281348221_Internet_Voting_Experiences_From_Five_Elections_in_Estonia

[2] S. Heiberg, T. Martens, P. Vinkel, and J. Willemson, "Improving the verifiability of the Estonian internet voting scheme," in *Electronic Voting* (Lecture notes in computer science). Cham, Switzerland: Springer, 2017, pp. 92–107, doi: 10.1007/978-3-319-52240-1_6.

[3] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, "Security analysis of the Estonian internet voting system," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2014, pp. 703–715, doi: 10.1145/2660267.2660315.

[4] A. Juvonen, "A framework for comparing the security of voting schemes," Master's thesis, Dept. Comput. Sci., Fac. Sci., Univ. Helsinki, Helsinki, Finland, 2019.

[5] ODIHR Election Expert Team. (Aug. 2023). *Estonia, Parliamentary Elections, 5 March 2023: Final Report*. [Online]. Available: https://www.osce.org/files/f/documents/f/f/551179_0.pdf

[6] Organization for Security and Co-operation in Europe. (Jun. 2025). *Estonia: Opinion on the Regulation of Internet Voting*. [Online]. Available: https://www.osce.org/files/f/documents/e/a/593435.pdf

[7] State Electoral Office of Estonia. (May 2024). *Elektroonilise Hletamise Ldraamistik Ja Selle Kasutamine Eesti Riiklikel Valimistel*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2011%20lisa%20%28IVXV%20EHS%20%C3%BCldraamistik%29.pdf

[8] (2025). *IVXV Online Voting System*. [Online]. Available: https://github.com/valimised/ivxv

[9] T. Treier and K. Düüna, "Identifying and solving a vulnerability in the Estonian internet voting process: Subverting ballot integrity without detection," *IEEE Access*, vol. 12, pp. 197766–197782, 2024, doi: 10.1109/ACCESS.2024.3521337.

[10] J. Oruaas and H. Pldmaa. (Jul. 2024). *Euroopa Parlamendi Elektroonilise Hletuse Toimingute Audit*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-08/Aruanne%20EP2024%20e-valimiste%20audit_0.pdf

[11] S. Dzieduszycka-Suinat et al., "The future of voting: End-to-end verifiable internet voting—Specification and feasibility assessment study," U.S. Vote Found., Arlington, VA, USA, Galois, Inc., Portland, OR, USA, Tech. Rep., Jul. 2015.

[12] R. L. Rivest, "On the notion of 'software independence' in voting systems," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 366, no. 1881, pp. 3759–3767, Aug. 2008, doi: 10.1098/rsta.2008.0149.

[13] S. Heiberg, K. Krips, and J. Willemson. (2020). *Planning the Next Steps for Estonian Internet Voting*. [Online]. Available: https://www.academia.edu/download/64628661/2020_E-Vote-ID_Taltech_Press.pdf#page=100

[14] M. Bernhard, J. Benaloh, J. A. Halderman, R. L. Rivest, P. Y. A. Ryan, P. B. Stark, V. Teague, P. L. Vora, and D. S. Wallach, "Public evidence from secret ballots," in *Electronic Voting*. Cham, Switzerland: Springer, 2017, pp. 84–109, doi: 10.1007/978-3-319-68687-5_6.

[15] D. Clarke and S. T. Ali, "End to end security is not enough," in *Security Protocols XXV*, F. Stajano, J. Anderson, B. Christianson, and V. Matyáš, Eds., Cham, Switzerland: Springer, 2017, pp. 260–267, doi: 10.1007/978-3-319-71075-4_29.

[16] (May 2023). *Elektroonilise Hääletamise Protsessi Auditeerimine*. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-05/L%C3%B5pparuanne%20RKV23%20EValimiste%20Audit.asice

[17] (2024). *Riigikogu Valimise Seadus*. [Online]. Available: https://www.riigiteataja.ee/akt/124052024010

[18] K. Krips and J. Willemson, "On practical aspects of coercion-resistant remote voting systems," in *Electronic Voting*, U. Serdült, and D. Duenas-Cid, Eds., Cham, Switzerland: Springer, 2019, pp. 216–232, doi: 10.1007/978-3-030-30625-0_14.

[19] R. Giustolisi, M. S. Garjan, and C. Schuermann, "Thwarting last-minute voter coercion," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 3423–3439, doi: 10.1109/SP54263.2024.00112.

[20] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, and C. Rosenthal, "Chaos engineering," *IEEE Softw.*, vol. 33, no. 3, pp. 35–41, May 2016, doi: 10.1109/MS.2016.60.

[21] Estonian Academy of Sciences, Cybersecurity Committee. (2024). *TA Küberturvalisuse Komisjoni Valimiste Tehnoloogia Riskianaliüüs 2024*. [Online]. Available: https://koodivaramu.eesti.ee/riigi-valimisteenistus/valimised-turve/-/blob/main/2024/TA-k%C3%BCberturvalisuse-komisjoni-valimiste-tehnoloogia-riskianal%C3%BC%C3%BCs-2024.pdf

[22] S. Baranski, B. Biedermann, and J. Ellul, "A trust-centric approach to quantifying maturity and security in internet voting protocols," 2024, *arXiv.2412.10611*.

[23] J. Müller, "Breaking and fixing vote privacy of the Estonian e-voting protocol IVXV," in *Proc. Int. Workshops Financial Cryptography Data Secur.*, S. Matsuo, L. Gudgeon, A. Klages-Mundt, D. Perez Hernandez, S. Werner, T. Haines, A. Essex, A. Bracciali, and M. Sala, Eds., Cham, Switzerland: Springer, 2023, pp. 325–334, doi: 10.1007/978-3-031-32415-4_22.

[24] E. G. Boone, "Coercion-resistance and end-to-end verifiability in the Estonian and Swiss CHVote 2.0 evoting systems," Master's thesis, Dept. Inform., Univ. Oslo, Oslo, Norway, 2022.

[25] A. Sutopo, T. Haines, and P. Roenne, "On the auditability of the Estonian IVXV system," in *Proc. Int. Workshops Financial Cryptography Data Secur.*, A. Essex, S. Matsuo, O. Kulyk, L. Gudgeon, A. Klages-Mundt, D. Perez, S. Werner, A. Bracciali, and G. Goodel, Eds., Cham, Switzerland: Springer, 2024, pp. 19–33, doi: 10.1007/978-3-031-48806-1_2.

[26] T. Finogina, J. Cucurull Juan, and N. Costa, "Selective comparison of verifiable online voting systems," *Secur. Privacy*, vol. 7, no. 5, p. 394, Sep. 2024, doi: 10.1002/spy2.394.

[27] K. Krips, N. Snetkov, J. Vakarjuk, and J. Willemson, "Trust assumptions in voting systems," in *Proc. Int. Workshops Comput. Secur.* Switzerland: Springer, 2024, pp. 309–329, doi: 10.1007/978-3-031-54129-2_18.

[28] F. Moser et al., "A study of mechanisms for end-to-end verifiable online voting," Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany, Tech. Rep., 2024.

[1] OpenAI. ChatGPT (GPT-5 Thinking), 2025. https://www.openai.com/

[29] J. C. L. A. de Farias, A. Carniel, J. de Melo Bezerra, and C. M. Hirata, "Approach based on STPA extended with STRIDE and LINDDUN, and blockchain to develop a mission-critical e-voting system," *J. Inf. Secur. Appl.*, vol. 81, Mar. 2024, Art. no. 103715. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212624000188

[30] O. Pereira and V. Teague. (Jul. 2019). *Report on the SwissPost-Scytl E-Voting System, Trusted-Server Version*. [Online]. Available: https://www.bk.admin.ch/dam/bk/de/dokumente/pore/E-Voting_Report_Pereira_Teague_Juli%202019.pdf.download.pdf/E-Voting_Report_Pereira_Teague_Juli%202019.pdf

[31] T. Haines, S. J. Lewis, O. Pereira, and V. Teague, "How not to prove your election outcome," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 644–660, doi: 10.1109/SP40000.2020.00048.

[32] R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis. (Feb. 2023). *Re-Examination of the Swiss Post Internet Voting System*. [Online]. Available: https://www.bk.admin.ch/dam/bk/en/dokumente/pore/E_Voting/Examination_Reports_March2023/Scopes%201%20and%202%20Final%20Report%20and%20Addenda%20BFH%2023.02.2023.pdf.download.pdf/Scopes%201%20and%202%20Final%20Report%20and%20Addenda%20BFH%2023.02.2023.pdf

[33] P. A. Laplante and J. F. DeFranco, "Software engineering of safety-critical systems: Themes from practitioners," *IEEE Trans. Rel.*, vol. 66, no. 3, pp. 825–836, Sep. 2017, doi: 10.1109/TR.2017.2731953.

[34] *Efficient Verification Through the DO-178C Life Cycle*, document MC-WP-011, Rapita Systems Ltd., Dec. 2021. [Online]. Available: https://www.rapitasystems.com/downloads/efficient-verification-through-do-178c-life-cycle

[35] *Software Assurance and Software Safety Standard*, NASA, Washington, DC, USA, Sep. 2022. [Online]. Available: https://standards.nasa.gov/sites/default/files/standards/NASA/B/0/NASA-STD-87398-Revision-B.pdf

[36] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, and B. Monate, "Testing or formal verification: DO-178C alternatives and industrial experience," *IEEE Softw.*, vol. 30, no. 3, pp. 50–57, May 2013, doi: 10.1109/MS.2013.43.

[37] H. T. Lawless and H. Heymann. (2010). *Acceptance Testing*. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c96787b267b6128165df7f7f0ac4820c839312ce

[38] L. Wang and K. C. Tan, "Software testing for safety critical applications," *IEEE Instrum. Meas. Mag.*, vol. 8, no. 2, pp. 38–47, Jun. 2005, doi: 10.1109/MIM.2005.1438843.

[39] M. Mitev, "Measure twice, cut once: Acceptance testing," in *The Future of Software Quality Assurance*, 2020, pp. 93–104.

[40] K. Fowler, "Mission-critical and safety-critical development," *IEEE Instrum. Meas. Mag.*, vol. 7, no. 4, pp. 52–59, Dec. 2004, doi: 10.1109/MIM.2004.1383466.

[41] M. McGaley and J. P. Gibson, "Electronic voting: A safety critical system," B.Sc. Final Year Project, Dept. Comput. Sci., Nat. Univ. Ireland, Maynooth, Co. Kildare, Ireland, Mar. 2003.

[42] B. Douglass, "Safety-critical systems design," *Electron. Eng.*, vol. 70, no. 862, pp. 6–45, 1998. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ee8aa026160a1806ae5746e8a3c6154bc40a6c81

[43] A. J. Offutt, "A practical system for mutation testing: Help for the common programmer," in *Proc. Int. Test Conf.*, 1994, pp. 824–830, doi: 10.1109/test.1994.528535.

[44] State Electoral Office of Estonia. (May 2024). *IVXV Seadistuste Koostamise Juhend*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2012%20lisa%20%28IVXV-seadistuste%20koostejuhend%29.pdf

[45] (May 2024). *IVXV Arhitektuur*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2012%20lisa%20%28IVXV-arhitektuur%29.pdf

[46] (May 2024). *IVXV: E-Hletamise Ksiraamat*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-05/RVT%20Korraldus%20nr%2010%20lisa%20%28IVXV%20e-h%C3%A4%C3%A4letamise%20k%C3%A4siraamat%2008%29-1.pdf

[47] (May 2024). *IVXV Protokollide Kirjeldus*. [Online]. Available: https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2012%20lisa%20%28IVXV-protokollide%20kirjeldus%29.pdf

[48] SK ID Solutions. (2025). *Test Number for Automated Testing in Demo*. [Online]. Available: https://github.com/SK-EID/MID/wiki/Test-number-for-automated-testing-in-DEMO

TARVO TREIER was born in Võru, Estonia, in 1982. He received the B.Sc. and M.Sc. (cum laude) degrees in computer science from Tallinn University of Technology, Estonia, in 2005 and 2007, respectively, where he is currently pursuing the Ph.D. degree with the Department of Software Science.

Since 2013, he has been a Lecturer with the Department of Software Science, Tallinn University of Technology. He has also worked at several companies as a Software Engineer. With more than 20 years of experience in software engineering, he specializes in streamlining workflows and enhancing efficiency. His research interests include e-government, distributed systems, and distributed applications.

Appendix 3

III

T. Treier. Re-voting under surveillance: National eID transaction logs as a threat to coercion resistance in Estonian internet voting. In D. Duenas-Cid, P. Roenne, M. Volkamer, M. Blom, P. Gaudry, I. Borucki, L. Loeber, and A. Debant, editors, *Electronic Voting*, pages 191–207, Cham, 2025. Springer Nature Switzerland

Re-voting Under Surveillance: National eID Transaction Logs as a Threat to Coercion Resistance in Estonian Internet Voting

Tarvo Treier( )

Department of Software Science, Tallinn University of Technology, 12618 Tallinn, Estonia
tarvo.treier@taltech.ee

Abstract. Estonia's nationwide Internet-voting scheme relies on the state-mandated electronic identity (eID) infrastructure for strong voter authentication and qualified electronic signatures. A statutory safeguard against coercion is re-voting: a voter may cast multiple electronic ballots during the advance-voting period, with only the last one counted, and the number of ballots must remain secret. This expectation is violated by current eID audit practice: every signing transaction-including each vote-is irrevocably logged by the eID service provider and displayed to the credential holder. These logs reveal the exact count and timing of a voter's interactions with the voting system, compromising the intended secrecy of re-voting and enabling coercers to detect whether the voter has changed their choice. This paper presents a threat model and examines concrete design alternatives-such as persistent log filtering, dedicated voting credentials, and offline signing-analysing their respective trade-offs in security, usability, regulatory compliance, and system complexity. The findings demonstrate how well-intentioned components can interact to break coercion resistance through a metadata-based side-channel. The identified vulnerability has been responsibly disclosed to the relevant Estonian authorities. The case underlines the need for composition-aware risk assessment whenever election systems depend on external digital infrastructures.

Keywords: Internet voting · coercion resistance · side-channel · electronic identity · Estonia

1 Introduction

Internet voting (i-voting) allows citizens to cast their vote outside a traditional polling station, from any location with internet access. This increases convenience and may raise turnout, but it also introduces new challenges for election security, integrity, secrecy, and voter freedom. A particularly critical issue is the

D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 191–207, 2026.
https://doi.org/10.1007/978-3-032-05036-6_12

integration of national digital infrastructures-especially electronic identity (eID) systems-into the electoral process. While eID systems improve authentication and usability, their use in voting may introduce unforeseen security risks and goal conflicts.

1.1 Background and Motivation

Estonia is widely recognised as a pioneer of digital democracy, having introduced nationwide i-voting already in 2005 [3]. Its success is underpinned by a mandatory eID infrastructure [14]. The same infrastructure used by citizens for daily digital signatures and secure transactions has also enabled the move to online elections. However, this infrastructural trust may become a vulnerability if side effects compromise other critical goals.

One of the core features of Estonia's i-voting system is re-voting, which allows voters to cast multiple electronic ballots during the advance-voting period, with only the last one being counted. This mechanism is intended to enhance coercion resistance [10], allowing a voter to privately override a coerced vote [24]. Estonian law (§48^6(9) of the Riigikogu Election Act) requires that the system must not allow a voter to prove to anyone which of their electronic votes was counted [20]. This implies that it must be impossible to detect how many votes were cast or when. The Estonian Supreme Court has affirmed the central role of re-voting in safeguarding electoral freedom, calling it the only sufficiently effective measure for ensuring voter autonomy under electronic conditions [25].

Re-voting, therefore, must be more than a theoretical safeguard-it must work in practice. Its success depends in part on the secrecy of voting frequency and timing. A 2024 risk analysis by the Cybersecurity Committee of the Estonian Academy of Sciences identified 31 distinct risks in i-voting [4], none of which covered re-vote detection via eID logs.[1]

A public demonstration in March 2023 showed that combining an e-vote cryptogram, the eID transaction log and an official confirmation from the Electoral Service of whether the electronic vote was overridden can prove how an individual voted [19]. The Electoral Service responded that this proof is no stronger than photographing a paper ballot [12]. A lightning talk at E-Vote-ID 2023 reiterated the same risk and pointed to the eID transaction log as evidence [18]. None of these sources, however, analysed the log itself as an independent side-channel, mapped its legal implications, or compared mitigation options. The present paper fills that gap by presenting a threat model, demonstrating in practice how the eID log leaks re-voting, and comparing six mitigation families side-by-side.

[1] The author notified a member of the Cybersecurity Committee of the Estonian Academy of Sciences about the missing log-trace risk on 2 Oct 2024. At a meeting with the National Electoral Service on 4 Jun 2025, several committee members confirmed that mitigation work was under way. The accepted manuscript was forwarded to the Electoral Service on 24 Jun 2025.

1.2 Problem Statement

Classic side-channel attacks exploit physical characteristics of hardware-such as power use, timing patterns, or cache behaviour [11,26]. However, modern information systems may leak sensitive data through metadata produced for legitimate purposes. This paper refers to such leakage as a *metadata-based side-channel*.

In the Estonian case, the risk arises not from cryptographic flaws, but from the interplay of otherwise secure systems. Every qualified digital signature is logged by the eID service, including each e-vote. A typical log entry may read `Signing, 2025-03-01 14:02:17, Voting, OK`. These entries are visible to the credential holder and persist under long-term audit rules. Consequently, both voters and coercers may observe the number and timing of votes cast. Although no vote content is revealed, this metadata alone compromises re-voting secrecy and undermines receipt-freeness. Receipt-freeness means that an attacker cannot determine a voter's exact behaviour, which prevents them from coercing the voter by dictating a specific choice [10].

This represents a composition attack [8], where the combination of two secure components-the national eID transaction log and the re-voting protocol-produces an unintended privacy vulnerability. The metadata in eID transaction logs functions as a side-channel, revealing whether a voter has changed their vote.

While logs from the voting system itself are anonymised or destroyed according to electoral law, eID transaction logs are retained for at least ten years in line with trust service regulation. Thus, re-voting remains traceable long after the election concludes.

However, the presence of verifiable metadata alone presents a serious and independent threat to the effectiveness of re-voting. While a coercer can sometimes persuade a voter to photograph a paper ballot, the eID transaction log constitutes a markedly different form of evidence. It is produced automatically by a qualified trust-service provider, written once and retained for at least ten years, creating a persistent and highly credible record of the voter's actions. That official provenance gives coercers a systemic, low-cost channel for checking whether a re-vote occurred-a threat that scales far more easily than methods requiring physical presence or elaborate coordination.

Moreover, because the entries remain available for years, their mere perceived auditability may deter voters from casting a secret re-vote even if the coercer never consults them. As soon as the coercer has supervised the voter once-during an i-voting session-demanding a follow-up look at the transaction log after the election adds virtually no burden; the voter is already accustomed to the coercer's scrutiny.

1.3 Contribution

This study is based on the premise that re-voting is Estonia's primary coercion-resistance mechanism, and that its effectiveness hinges on concealing the number

and timing of voting actions. It is shown that transaction logs from the national eID system create a metadata-based side-channel that compromises this secrecy.

The contributions of this work are as follows:

- **Threat Model:** A model characterises log leakage and states the exact condition under which re-voting secrecy is violated.
- **Impact Assessment:** The effect of this side-channel on coercion resistance is evaluated from both technical and legal perspectives.
- **Mitigation Strategies:** Several design alternatives are compared-such as log filtering, separate voting credentials, and offline signing-and their trade-offs analysed.
- **Risk Taxonomy Extension:** An extension to the 2024 national risk register is proposed, adding a category for metadata-based side-channels and outlining recommendations for mitigation.

The Estonian case illustrates a broader lesson: coercion resistance can fail when component interactions are not subject to composition-aware risk assessment. This paper provides a foundation for such analysis in i-voting systems.

2 Background: Re-voting and eID Transaction Logs

Estonia's i-voting scheme depends on two components built for different purposes: the re-voting protocol and the nationwide eID ecosystem operated by qualified trust-service providers. Each component is effective for its original goal, yet their interaction creates the metadata-based side-channel analysed in Sect. 3. This section explains the two pillars and relates the Estonian setting to experiences in Norway and Switzerland.

2.1 Re-voting as a Safeguard Against Coercion

Estonian i-voting allows a voter to submit multiple electronic ballots during the advance-voting period; only the last ballot is counted [24]. The mechanism serves two purposes:

1. **Correction:** a coerced or erroneous vote can be privately overridden later.
2. **Deterrence:** a coercer can never be sure the supervised vote will remain final.

Section $48^6(9)$ of the *Riigikogu Election Act* bars a voter from proving which electronic ballot was counted [20]. The Estonian Supreme Court calls re-voting practically the only sufficiently effective safeguard of voter freedom in an uncontrolled environment (3-4-1-13-05, para 30) [25]. Foundational work on election security defines a hierarchy of goals: *receipt-freeness*-the voter cannot prove how they voted-and the stronger *coercion resistance*-an adversary cannot tell whether the voter obeyed a demand [10]. In the Estonian system, re-voting is presented as the primary means of approximating these goals. The safeguard works only while the number and timing of votes remain secret.

Prior research already questions whether re-voting is compatible with other goals. Pereira shows that a malicious client can silently replace the voter's final encrypted ballot, leaving the voter unsure which ballot is stored [16]. This paper exposes a complementary weakness-a coercer can detect re-voting via metadata in the eID transaction log-and compares concrete defences.

When nationwide i-voting started in 2005, voters had no convenient public view of signature logs, so re-voting was deemed effective even outside polling places. Over time, user-facing portals such as *MyID*[2] were launched, letting citizens browse every qualified signature they produced. This transparency created a metadata side-channel: a coercer who sees the log can confirm whether and when the voter re-voted. Section 3 analyses this channel.

2.2 The National eID Ecosystem and Its Transaction Logs

The Estonian eID ecosystem-ID Card, Mobile-ID and Smart-ID-is widely used in banking and public e-services [14]. Each qualified signature or authentication produces a transaction-log record that voters can view in *MyID*. Crucially, these logs are maintained by the trust-service provider and cannot be altered or deleted by the credential holder-the provider may only append new records-thereby preserving their integrity for audit purposes. The logs serve multiple goals:

- **User Transparency:** Citizens can detect misuse of their identity.
- **Non-repudiation Evidence:** European Telecommunications Standards Institute (ETSI) EN 319 401 §7.10 mandates log retention [6].
- **Incident Response:** ETSI EN 319 401 §7.9.1 requires anomaly detection [6].
- **Regulatory Compliance:** Article 24 of Electronic Identification and Trust Services (eIDAS) Regulation obliges trust-service providers to implement security measures and report incidents [5].
- **Ten-Year Retention:** The service provider's privacy policy states that trust-service logs are kept for at least ten years to prove service correctness [22].

A simplified entry for an i-voting action may read:

```
Signing   2025-03-01 14:02:17   Voting   OK
```

The log reveals no ballot content, but does disclose:

- the exact number of vote-related signatures, and
- timestamps with one-second precision for each signature.

Because the audit trail is visible to the voter-and to anyone who can compel its disclosure-the metadata acts as a de facto receipt showing whether and when a re-vote occurred. Election-system logs, by contrast, must be anonymised or destroyed, creating a systemic inconsistency that enables the side-channel analysed next.

[2] SK ID Solutions AS, https://myid.skidsolutions.eu/en/.

2.3 International Parallels

Other countries have found that transparency artefacts can be repurposed as receipts.

- **Norway (Return Codes):** During the 2011 and 2013 i-voting pilots, each electronically cast ballot triggered an ordinary SMS whose return code matched the personal code sheet mailed to the voter in advance. As in Estonia, a voter could cast a new electronic ballot to override a coerced one, and every re-vote generated a fresh SMS. Although the message thread could in principle be deleted, coercion resistance still depended on the voter remembering to remove the codes before any potential coercer inspected the phone. Researchers warned that such codes "simplify the task of coercers and vote buyers" [1]. The government discontinued the pilot in 2014, citing security concerns and waning political support [15].
- **Switzerland (sVote Receipt):** The sVote platform returned a digitally signed package containing the encrypted ballot and a return code that voters could compare with a pre-printed paper sheet. A subsequent technical audit concluded that the same package allows the client to prove that a particular vote was cast, because the server signature binds the code to the encrypted ballot and is publicly verifiable [17]. Further cryptographic flaws were shown to let manipulated votes pass verification, and-because of those findings-no version of sVote was used in either the May 2019 Swiss referendum or the federal elections of October 2019 [9].

Sewell and Vitek note that assembling a system from partially trustworthy components is safe only if their interactions are encapsulated by a wrapper mechanism [21]. The Norwegian and Swiss cases illustrate that un-encapsulated interactions can create unexpected side-channels. In Estonia, no such wrapper exists between eID transaction logs and re-voting, so neutral metadata can function as a receipt and undermine coercion resistance. The next section analyses this side-channel in depth.

3 Metadata-Based Side-Channel Analysis

This section shows how the re-voting protocol R and the eID transaction log L-each designed for a different purpose-interact to create a metadata side-channel that undermines coercion resistance. First, a concise threat model is presented. Second, the impact of composition on security properties is examined. Finally, residual risk is assessed even in the presence of the paper-ballot fallback.

3.1 Threat Model and Attack Scenario

The leakage threat is presented precisely: the model specifies exactly what metadata an adversary can observe and how that observation leads to a coercion decision, without introducing additional cryptographic assumptions.

Participants. A single voter V uses the re-voting protocol R. A coercer C can supervise one voting session and later compel V to display the voter-accessible *MyID* view of the log L_V. The qualified trust-service provider (TSP) appends log records and offers voters no means to alter or delete existing ones; any modification would require collusion with, or compromise of, the provider itself.

Log Example. Each e-vote appends an immutable entry such as

$\langle type = \mathsf{Signing}, time = \mathsf{2025\text{-}03\text{-}01\ 14{:}02{:}17}, service = \mathsf{Voting}, result = \mathsf{OK} \rangle$.

Definition 1 (Log-leakage). *Let $L_V^{\mathsf{vote}} \subseteq L_V$ be the subsequence of vote-related entries. The leakage function*

$$\mathcal{F}(L_V^{\mathsf{vote}}) = \langle n, t_1, \ldots, t_n \rangle, \qquad t_1 < \cdots < t_n$$

returns the number n of ballots cast by V and the corresponding one-second-precision timestamps $t_1, \ldots, t_n$.

Attack Steps.

- **S1** At the start of the supervised session C instructs V to open *MyID*. V displays the current log view, revealing $k = |\mathcal{F}(L_V^{\mathsf{vote}})|$, the number of vote entries recorded so far.[3]
- **S2** A ballot is then cast under coercer supervision at time t_{k+1}. C orders V not to vote again.
- **S3** After the voting period closes C compels V to reopen *MyID*. C observes $\mathcal{F}(L_V^{\mathsf{vote}}) = \langle n, t'_1, \ldots, t'_n \rangle$.
- **S4** A re-vote is deemed to have occurred *iff* $n > k{+}1$. Otherwise the supervised ballot is accepted as potentially counted.

Counting the entries alone suffices; timestamps merely indicate when any unsupervised vote was cast. Because the log is immutable and voter-accessible, the two cases can be distinguished with certainty-no cryptanalysis or privileged system access is required. Merely knowing that such verification is possible may deter some voters from exercising their legal right to re-vote.

3.2 Breakdown of Security Properties Under Composition

Voter log L_V: Append-only, immutable, and reveals $\mathcal{F}(L_V)$ but no ballot content.

Re-vote R: Counts only the last ballot and assumes that $\mathcal{F}(L_V)$ remains hidden.

Composition: Once combined, the metadata becomes a verifiable receipt; both coercion resistance and receipt-freeness collapse.

Thus an adversary needs only the voter-specific log view together with the pre-coercion count k to decide whether a re-vote occurred; timestamps are optional corroborating evidence.

[3] A *minimal-assumption variant* needs only the supervised timestamp $t_{sup} = t_{k+1}$; re-voting is detected as soon as any later entry appears.

3.3 Paper-Ballot Override: Partial Remedy

Estonian law allows a voter to annul all prior electronic votes by casting a paper ballot in person on election day. This serves as a last-resort defence, yet its effectiveness against a determined coercer is mixed.

- **Extended Exposure:** Pressure can persist until the final day, because the coercer needs to influence only one physical act-the trip to a polling station.
- **Visibility and social pressure:** Travelling to a polling place is a public act. Informal social monitoring (e.g. by family or neighbours) can make a secret visit risky, without requiring continuous digital surveillance.
- **Post-election Record:** After polling day any voter may request an official certificate from the National Electoral Service stating whether the final counted ballot was electronic or paper.[4] A coercer could compel the voter to obtain and reveal this confirmation, providing a definitive test for a paper-ballot override. Although mass requests would be conspicuous, the mere possibility further reinforces the deterrent.

Paper ballots therefore mitigate, but do not eliminate, the incentive to coerce.

3.4 Residual Risk Assessment

Although no ballot content is revealed, the metadata alone serves as a credible receipt. This is especially damaging because coercion itself leaves no corresponding trace in the election system, undermining the deterrent effect of re-voting. From an adversary's viewpoint the cost is low: the pre-coercion count k and a single post-election inspection of the voter-accessible log are sufficient. Coercion resistance in the Estonian scheme thus hinges on hiding voting frequency-an assumption invalidated by current e-ID audit practice, which exposes $\mathcal{F}(L_V^{\mathsf{vote}})$ to the credential holder. Breaking the data flow from the voter-accessible log to the coercer is therefore essential; Sect. 4 explores concrete counter-measures.

4 Mitigation Strategies for the Metadata-Based Side-Channel

This section surveys technical and procedural measures that could neutralise, dampen, or compensate for the side-channel identified in Sect. 3. Five classical counter-measure classes-Hiding, Isolation, Leakage Dampening, Masking, and Encryption-are adopted from the taxonomy of Autio et al. [23]. For each class, one concrete design is sketched, its strengths and weaknesses are outlined, and a qualitative rating (High/Medium/Low) is given along three dimensions: effectiveness, deployability, and regulatory fit.

[4] A publicly available video demonstration illustrates how such a request and encrypted reply are obtained [19].

4.1 Hiding: Persistent Log Filter

A persistent filter would hide i-voting entries from the default *MyID* web view while retaining them on the server for audit purposes.

Advantages: Vote-related metadata is no longer visible online but remains accessible in a physical service centre upon request; audit staff retain full data.
Drawbacks: Transparency decreases, as some voters may mistrust a portal that no longer shows a "complete" history. Inspecting hidden entries requires visiting a service desk in person, adding mild inconvenience.
Effectiveness - High: Online leakage is blocked for routine coercion scenarios.
Deployability - High: Only front-end and API-level access-control changes are required.
Regulatory fit - High: ETSI and eIDAS retention rules are satisfied because no entries are deleted.

The filter therefore acts as a wrapper that interrupts the information flow between the trust-service database and routine users while preserving official access paths.

4.2 Isolation (A): Separate Voting Credential

A dedicated certificate issued solely for elections would have its own PKI hierarchy and separate logs.

Advantages: Election signatures never appear in the general e-ID log, so coercers learn nothing from that source.
Drawbacks: Requires a large roll-out effort, new middleware, and extra credential management, and may require new hardware or separate certificates on some devices.
Effectiveness - High: Leakage is eliminated at the source.
Deployability - Low: New PKI, software, and registration flows are needed.
Regulatory fit - High: Election logs fall outside commercial trust-service regulation, easing ten-year retention constraints.

The dedicated credential itself acts as a wrapper that cleanly separates election traffic from everyday e-ID usage. A similar approach is used in the *Selections* voting system [2].

4.3 Isolation (B): Offline Signing with the ID Card

A voting client could generate the signature locally inside the ID card chip and transfer only the sealed ballot, bypassing the remote signing service.

Advantages: No vote-related entry is written to the trust-service log, eliminating the side-channel without extra credentials.
Drawbacks: Works only for ID cards because Mobile-ID and Smart-ID rely on remote server-side signing, not local private keys. Requires new middleware and a wide software rollout. Immediate signature validation would be replaced by delayed verification, increasing the risk of undetected invalid signatures during voting.
Effectiveness - Medium: Central log leakage is completely avoided only for ID cards.
Deployability - Medium: New software is needed, but no PKI overhaul is required.
Regulatory fit - High: ETSI logging rules apply only to qualified remote signatures, not to local card operations.

The local signing client operates as a wrapper between the ID card chip and the voting server, eliminating remote log entries.

4.4 Leakage Dampening: User-Controlled Log Hiding

The portal could allow voters to hide specific entries from their online view after login.

Advantages: Quick to implement; empowers privacy-conscious voters by giving them direct control.
Drawbacks: Protection is entirely timing-dependent. The voter must hide the entry immediately after re-voting; any delay allows a coercer to compel the voter to show the log and detect the additional record. *The hide operation is one-way: once an entry is hidden, it cannot be unhidden online.*
Effectiveness - Medium: A one-shot defence that succeeds only if the voter hides entries before the coercer's next inspection.
Deployability - High: Requires front-end changes (e.g., a "hide" button) and corresponding API-level filtering.
Regulatory fit - High: Server logs remain intact, satisfying retention rules.

Here the voter interface itself acts as a wrapper, giving discretionary control over what metadata may leak. A comparable situation arose in Norway's 20112013 i-voting pilots [1], where return codes remained on voters' phones unless actively deleted.

4.5 Masking: Dummy Log Entries

The trust-service provider could inject realistic, cryptographically signed dummy entries mimicking voting signatures, obfuscating the real pattern.

Advantages: Even a visible log becomes ambiguous.
Drawbacks: Dummy management is complex and may pose legal risks if fake entries are challenged.
Effectiveness - Medium: Provides plausible deniability but may be overcome by sophisticated adversaries.
Deployability - Low: Requires dummy generation, separate signing keys, and retention policies.
Regulatory fit - Low: Artificial entries undermine evidential reliability.

The dummy-entry generator serves as a wrapper, blurring the leakage channel rather than sealing it. A similar obfuscation strategy is employed in the *VoteAgain* system [13], which introduces dummy ballots to obscure voting patterns.

4.6 Encryption: Controlled Log Access

Vote-related entries could be encrypted with a dual-key scheme; decryption would require the joint presence of the voter and a service operator at a physical office.

Advantages: Remote coercers cannot read the log; audit access remains possible.
Drawbacks: Requires new hardware security modules, procedures, and joint in-person sessions with operators, creating inconvenience. Effective only if the persistent log filter is already in place.
Effectiveness - High: Online leakage is blocked completely.
Deployability - Low: Significant infrastructure changes and new workflows are required.
Regulatory fit - High: Logs remain intact and auditable; only routine visibility is curtailed.

The dual-key encryption scheme and physical service point form a wrapper that blocks automated access while preserving auditable evidence. This mirrors "break-glass" protocols in healthcare, where dual control is required to access sensitive records [7].

4.7 Comparative Summary of Mitigation Options

The triplet after each option reports *Effectiveness* | *Deployability* | *Regulatory fit* (H = High, M = Medium, L = Low).

Persistent log filter - H|H|H: Quick fix with limited transparency loss.

Separate voting credential	**H|L|H:**	Strongest long-term solution; significant deployment burden.
Offline signing	**M|M|H:**	Strong for ID card users; excludes mobile credentials.
User-controlled hiding	**M|H|H:**	Inexpensive to implement, but shifts responsibility to voters.
Dummy entries	**M|L|L:**	Adds noise but weakens audit evidence.
Controlled log access	**H|L|H:**	Robust protection; costly and complex.

A phased response appears pragmatic. Introducing a persistent front-end filter offers quick and inexpensive mitigation against routine coercion. In parallel, exploring a dedicated voting credential provides a structurally stronger long-term solution, cleanly separating electoral activity from general-purpose e-ID usage.

5 Discussion

This section interprets the technical findings in a broader socio-legal context and reflects on the limitations of the present study. Five themes are addressed: the central finding, legal inconsistency, the interplay of verifiability and secrecy, threats to validity, and implications for future work.

5.1 Central Finding: Composition Matters

The analysis shows that coercion resistance in Estonian i-voting hinges on the secrecy of voting frequency. That secrecy collapses when the national eID transaction log is combined with the re-voting protocol, even though both components are sound in isolation. The result is a metadata-based receipt that a coercer can exploit without breaking cryptography or gaining privileged access. Similar composition pitfalls have appeared in Norway and Switzerland, where otherwise benign transparency artefacts became coercion tools. The lesson is clear: security claims must be re-evaluated whenever an election system is embedded in a wider digital infrastructure.

5.2 Legal Inconsistency: Two Retention Regimes

Section $48^6(67)$ of the *Riigikogu Election Act* requires the election service to erase or anonymise personal log data once the count is complete and the result certified. By contrast, Article 24 (2)(h) of the eIDAS Regulation obliges a qualified trust-service provider (TSP) to "*record and keep accessible, for as long as necessary after the activities of the provider have ceased, all relevant information concerning data issued and received … for the purpose of providing evidence in legal proceedings and for the purpose of ensuring continuity of the service*" [5].

ETSI EN 319 401 repeats the same principle in Clause 7.10 *Collection of evidence*: "*The TSP shall record and keep accessible for an appropriate period of time, including after the activities of the TSP have ceased, all relevant information concerning data issued and received ...*" [6].

The qualified TSP implements these texts through its privacy policy, which retains audit data related to qualified electronic signatures for at least ten years [22].

As a result, the same electronic ballot falls under two different retention rules: the election subsystem must purge identifying data soon after polling day, whereas the remote-signing subsystem keeps a linkable audit trail for a decade or more. This clash jeopardises the Election Act's intent that the system must not enable a voter to demonstrate which electronic ballot was ultimately counted.

Reconciling the regimes will require either (i) a statutory carve-out that explicitly exempts election signatures from the election-log deletion rule, or (ii) a technical wrapper that keeps the long-term eID audit trail hidden from routine disclosure while still fulfilling the provider's retention duty.

5.3 The Interplay of Verifiability and Secrecy

Fixing the eID log leak is a necessary first step, but it does not resolve the broader tension between verifiability and secrecy. This is illustrated by proposals for a public bulletin board (PBB)-a transparency mechanism currently absent from Estonian i-voting.

A PBB's primary goal is universal verifiability: publishing all received ballots (or their cryptographic checksums) allows voters and observers to verify that no ballots were dropped or altered. However, a PBB that explicitly marks each voter's final ballot [16] achieves verifiability at the cost of re-voting secrecy, creating an explicit receipt.

A checksum-only PBB is a privacy-preserving alternative avoiding the explicit receipt problem. However, this PBB alone cannot prevent a malicious client from tricking the voter into signing a substituted final ballot, as Pereira shows [16].

Thus, a holistic solution requires architectural layering: the system must first guarantee what-you-see-is-what-you-sign to defend against client-side attacks. Only then can a checksum-only PBB meaningfully defend against a malicious server. Sealing the existing eID log leak is a prerequisite for this entire architecture to become coercion-resistant.

5.4 Threats to Validity

Four limitations should be noted.

Scope: The study focuses on national eID transaction logs and does not analyse other potential side-channels such as network traffic or client logs.

Behavioural Data: No large-scale empirical survey of voter responses to log visibility was conducted; deterrence is inferred from plausibility and precedent.

Dynamic Environment: Regulatory frameworks and platform features evolve; mitigation proposals must therefore be revisited before deployment.

Legal interpretations: The author is not a legal expert. All regulatory interpretations in this paper reflect a technical perspective and should be independently verified from a legal viewpoint before policy implementation.

These caveats do not negate the main argument but indicate areas for further research.

5.5 Implications for Future Work

Future research could quantify the behavioural impact of log visibility through controlled user studies. Legal scholars may examine how trust-service regulation could accommodate election-specific secrecy requirements without weakening auditability in other domains. Finally, system designers should incorporate composition-aware risk assessment into standard election-technology life cycles.

6 Conclusion and Recommendations

This paper has shown that the national eID transaction log exposes the number and timing of electronic ballots, thereby defeating the secrecy on which Estonia's re-voting safeguard depends. The vulnerability is not rooted in cryptographic failure but arises from the composition of two otherwise secure systems: the re-voting protocol and the long-term audit trail mandated for qualified electronic signatures. Because the leakage involves only metadata, it persists even if ballot content remains perfectly encrypted. The result is a credible receipt exploitable for coercion at negligible technical cost.

6.1 Actionable Recommendations

To mitigate the issue, four concrete actions are proposed, each varying in complexity and timeframe:

1. **Immediate Mitigation Persistent Log Filter:** Hide vote-related entries in the web portal by default, retaining physical access for voters who need it. This change is inexpensive, complies with eIDAS retention rules, and blocks the routine side-channel for *all* credential types.
2. **Long-Term Option Separate Voting Credential:** Issue an election-specific certificate or token with its own audit trail, isolating voting activity from everyday eID use without restricting voter access to their data.

3. **Regulatory Alignment:** Amend either trust-service regulation or Estonia's election law to resolve the conflict between the decade-long retention of eID transaction data and short-lived election logs.
4. **Composition-Aware Risk Analysis:** Make cross-system interaction analysis mandatory whenever the Estonian i-voting scheme or its supporting infrastructures undergo modification, explicitly addressing the risks identified by composition analysis.

Successful implementation will require coordination among election officials, trust-service providers, regulators, legislators, and voters.

6.2 Final Remark

Democratic legitimacy depends not only on cryptographic soundness, but also on the social realities of voter autonomy. As digital infrastructures expand, election systems must be assessed in the context of these broader ecosystems. Only ongoing, multidisciplinary analysis-spanning technical, legal, and social domains-can prevent well-intentioned features from interacting in harmful ways.

Acknowledgments. The author used OpenAI's ChatGPT(OpenAI. ChatGPT (model o3), 2025. https://www.openai.com/) to improve language and structure. All research ideas, analysis and conclusions are solely the author's.

References

1. Barrat, J., Chevalier, M., Goldsmith, B., Jandura, D., Turner, J., Sharma, R.: Internet voting and individual verifiability: the Norwegian return codes. In: 5th International Conference on Electronic Voting 2012 (EVOTE2012), pp. 35–45. Gesellschaft für Informatik e.V., Bonn (2012). https://dl.gi.de/items/e9eeff35-f08c-4e0f-ab6d-47bbc8a5ea6e
2. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27576-0_4
3. Drechsler, W., Madise, Ü.: Electronic voting in Estonia. In: Kersting, N., Baldersheim, H. (eds.) Electronic Voting and Democracy: A Comparative Analysis, pp. 97–108. Palgrave Macmillan UK, London (2004). https://doi.org/10.1057/9780230523531_6
4. Estonian Academy of Sciences, Cybersecurity Committee: TA küberturvalisuse komisjoni valimiste tehnoloogia riskianalüüs 2024. https://koodivaramu.eesti.ee/riigi-valimisteenistus/valimised-turve/-/blob/main/2024/TA-k%C3%BCberturvalisuse-komisjoni-valimiste-tehnoloogia-riskianal%C3%BC%C3%BCs-2024.pdf (2024)
5. European Parliament and Council: Regulation (EU) no 910/2014 of the European parliament and of the council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec. https://eur-lex.europa.eu/eli/reg/2014/910 (2014)

6. European Telecommunications Standards Institute: ETSI EN 319 401 v3.1.1: Electronic Signatures and Infrastructures (ESI); General Policy Requirements for Trust Service Providers. https://www.etsi.org/deliver/etsi_en/319400_319499/319401/03.01.01_60/en_319401v030101p.pdf (2024)
7. Ferreira, A., et al.: How to break access control in a controlled manner. In: 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06), pp. 847–854 (2006). https://doi.org/10.1109/CBMS.2006.95
8. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 265–273. KDD '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1401890.1401926
9. Haines, T., Lewis, S.J., Pereira, O., Teague, V.: How not to prove your election outcome. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 644–660 (2020). https://doi.org/10.1109/SP40000.2020.00048
10. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70. WPES '05, Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1102199.1102213
11. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
12. Liive, R.: Valimisteenistus seletab lahti, miks digiaktivisti väiteid e-hääle salajasuse murdmise osas ei vasta tõele. Geenius Meedia (2023). https://digi.geenius.ee/rubriik/uudis/valimisteenistus-seletab-lahti-miks-digiaktivisti-vaiteid-e-haale-salajasuse.murdmise-osas-ei-vasta-toele/
13. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: a scalable coercion-resistant voting system. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 1553–1570. USENIX Association (2020). https://www.usenix.org/conference/usenixsecurity20/presentation/lueks
14. Martens, T.: Electronic identity management in Estonia between market and state governance. Ident. Info. Soc. **3**(1), 213–233 (2010). https://doi.org/10.1007/s12394-010-0044-0
15. Norwegian Ministry of Local Government and Modernisation: Internet voting pilot to be discontinued. https://www.regjeringen.no/en/historical-archive/solbergs-government/Ministries/kmd/press-releases/2014/Internet-voting-pilot-to-be-discontinued/id764300/ (2014)
16. Pereira, O.: Individual verifiability and revoting in the Estonian internet voting system. In: Matsuo, S., Gudgeon, L., Klages-Mundt, A., Perez Hernandez, D., Werner, S., Haines, T., Essex, A., Bracciali, A., Sala, M. (eds.) Financial Cryptography and Data Security. FC 2022 International Workshops, pp. 315–324. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-32415-4_21
17. Pereira, O., Teague, V.: Report on the SwissPost-Scytl e-voting system, trusted-server version. https://www.bk.admin.ch/dam/bk/de/dokumente/pore/E-Voting_Report_Pereira_Teague_Juli%202019.pdf.download.pdf/E-Voting_Report_Pereira_Teague_Juli%202019.pdf (2019)
18. Põder, M.: Hard evidence of an e-vote in the IVXV protocol. Lightning talk, E-Vote-ID 2023 (2023). https://infoaed.ee/proof2023/
19. Põder, M.: Kuidas tõendada oma e-häält? https://youtu.be/vWMaoqPY2-M (2023). YouTube

20. Riigikogu: Riigikogu valimise seadus. https://www.riigiteataja.ee/akt/124052024010 (2024). English Translation available
21. Sewell, P., Vitek, J.: Secure composition of insecure components. In: Proceedings of the 12th IEEE Computer Security Foundations Workshop. pp. 136–150 (1999). https://doi.org/10.1109/CSFW.1999.779769
22. SK ID Solutions AS: Principles of processing personal data (privacy policy). https://www.skidsolutions.eu/upload/files/SK-POPPD-EN-CURRENT.pdf (2022)
23. Spence, A., Bangay, S.: Security beyond cybersecurity: side-channel attacks against non-cyber systems and their countermeasures. Int. J. Inf. Secur. , 1–17 (2021). https://doi.org/10.1007/s10207-021-00563-6
24. State Electoral Office of Estonia: Elektroonilise hääletamise üldraamistik ja selle kasutamine Eesti riiklikel valimistel (2024). https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2011%20lisa%20%28IVXV%20EHS%20%C3%BCldraamistik%29.pdf
25. Supreme Court of Estonia: Riigikohtu põhiseaduslikkuse järelevalve kolleegiumi otsus 3-4-1-13-05. https://www.riigikohus.ee/et/lahendid?asjaNr=3-4-1-13-05 (2005). English translation available
26. Yarom, Y., Falkner, K.: Flush+reload: a high resolution, low noise, L3 cache side-channel attack. In: 23rd USENIX Security Symposium (USENIX Security 14), pp. 719–732. USENIX Association, San Diego, CA (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom

Appendix 4

Full Requirement–Fault Mapping Data

This appendix provides the extended baseline artefacts used in the Phase A methodology reported in Publication II. It complements the compact summary in Chapter 3 by presenting (i) the full fault catalogue, (ii) the full functional requirements list, and (iii) the dissertation-level requirement–fault mapping used for coverage planning. The appendix is intended as a traceability aid. It documents the logic of the current corpus and identifies where coverage remains direct, indirect, or currently absent. Where the published article presents only a compact subset and representative gaps, the appendix consolidates the broader mapping logic used in this dissertation.

Detection oracles

The primary audit oracles used in the mapping are the following:

O_1 **Set consistency:** No unexpected additions, removals, or duplicates in defined sets.

O_2 **Last-vote invariant:** Only the latest eligible e-vote per voter remains valid, unless invalidated by a paper vote.

O_3 **Paper override:** All e-votes of a paper voter are invalidated.

O_4 **District mapping preservation:** Anonymised outputs preserve correct electoral-district attributes.

O_5 **Signature and packaging validity:** All signatures and container structures match their specifications.

O_6 **Cross-artefact consistency:** Values in related artefacts match exactly across linked inputs and outputs.

Full fault scenarios (F_1–F_{37})

Table 6: Full fault catalogue used in the Phase A methodology.

Fault ID	Fault scenario description	Expected oracle(s)
Input errors (i-BB contents and metadata)		
F_1	A signed ZIP archive hash does not match the actual archive.	O_5
F_2	Missing per-ballot subfiles (e.g., `version`, `tspreg`, `ocsp`, `bdoc`).	O_5
F_3	Subfiles are swapped between ballots.	O_5, O_6
F_4	A BDOC container is oversized.	O_5
F_5	Missing cryptograms.	O_5
F_6	Empty cryptograms.	O_5

Fault ID	Fault scenario description	Expected oracle(s)
F_7	Invalid BDOC signature.	O_5, O_6
F_8	BDOC signature time differs from the registration timestamp.	O_6
F_9	Confirmations are signed with the wrong registration-service certificate.	O_5
F_{10}	E-votes cast outside the legal window (before start).	O_6
F_{11}	E-votes cast outside the legal window (after end).	O_6
F_{12}	Multiple identical cryptograms with the same confirmation.	O_1
F_{13}	Multiple identical cryptograms with different confirmations.	O_6
F_{14}	Mismatches between the registration list and i-BB contents.	O_6
F_{15}	E-votes from ineligible voters (not on valid lists).	O_6
F_{16}	Malformed cryptogram inside the BDOC.	O_5
F_{17}	BDOC profile mismatch (e.g., signed with the wrong profile relative to `BDOC_TS`).	O_5
Errors in other input lists		
F_{18}	Voter lists include missing or invalid signatures.	O_5
F_{19}	Voter lists have invalid file structure or contents.	O_5
F_{20}	District definitions include missing or invalid signatures.	O_5
F_{21}	District definitions have invalid file structure or contents.	O_5
F_{22}	Cancellation lists include missing or invalid signatures.	O_5
F_{23}	Cancellation lists have invalid file structure or contents.	O_5
F_{24}	Signed ZIP archive hash for the registration bundle does not match the actual archive.	O_6
Output-related scenarios prior to mixing		
F_{25}	Incorrect electoral-district assignment in output.	O_4
F_{26}	Addition of a new unique e-vote not present in the i-BB.	O_1
$F_{26\text{-}2}$	Unique ballot addition with supporting log edits.	O_1, O_6
F_{27}	Insertion of a duplicate e-vote.	O_1
F_{28}	Removal of an existing valid e-vote.	O_1
F_{29}	Replacement by a new unique e-vote.	O_1
F_{30}	Replacement by a duplicate of another e-vote.	O_1
F_{31}	Swapping an overridden (non-latest) e-vote's cryptogram with another voter's latest cryptogram from the same district.	O_6
F_{32}	Swapping an overridden (non-latest) e-vote's cryptogram with another voter's latest cryptogram from a different district.	O_4, O_6
F_{33}	Altering the order of a multi-voter's e-votes.	O_2
F_{34}	Altering the timestamps of a multi-voter's e-votes.	O_2
F_{35}	Wrongful invalidation of a valid e-vote.	O_2
F_{36}	Wrongful validation of a non-latest e-vote.	O_2
F_{37}	Failure to invalidate a voter's latest e-vote despite a paper vote.	O_3

Full functional requirements (R_1–R_{19})

Table 7: Functional requirements consolidated for the processing application in Publication II.

Req. ID	Category	Requirement description
R_1	Initialization & input	Use a valid, signed election configuration file.
R_2	Initialization & input	Ensure input readability and integrity pre-checks before processing.
R_3	Integrity & consistency	Validate each e-vote's digital signature.
R_4	Integrity & consistency	Verify the presence of a registration-service timestamp for each vote.
R_5	Integrity & consistency	Check consistency between collection and registration services.
R_6	Integrity & consistency	Detect and report inconsistencies between those services.
R_7	Integrity & consistency	Verify timestamp and certificate-status formats against protocol.
R_8	Identity & containers	Verify that the voter appears on the valid list.
R_9	Identity & containers	Validate the BDOC container profile and structure.
R_{10}	Identity & containers	Verify the correct naming of the encrypted ballot within the container.
R_{11}	Selection & cleansing	Identify re-votes and retain the latest submission.
R_{12}	Selection & cleansing	Maintain the linkage between voter identity and the last e-vote.
R_{13}	Selection & cleansing	Invalidate e-votes for voters who also cast a paper ballot.
R_{14}	Selection & cleansing	Ensure anonymisation occurs only after cleansing has completed.
R_{15}	Anonymisation & output	Anonymise e-votes while preserving necessary district attributes.
R_{16}	Anonymisation & output	Separate cryptograms from signatures as part of anonymisation.
R_{17}	Anonymisation & output	Output anonymised e-votes in the correct documented format.
R_{18}	Lists & logs	Generate both human-readable (PDF) and machine-readable lists of e-voters.
R_{19}	Lists & logs	Produce activity and error logs.

Requirement–fault mapping

Table 8: Dissertation-level requirement–fault mapping used for Phase A coverage planning.

Req. ID	Mapped fault scenarios	Coverage status and remarks
R_1	None currently mapped	**Gap.** A separate test case would be needed for invalid configuration, configuration-election mismatch, or configuration-signature failure.
R_2	$F_1, F_2, F_3, F_5, F_6, F_7, F_{16}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, F_{23}, F_{24}$	**Covered.** Input readability and integrity are exercised through packaging and structural anomalies.
R_3	F_2, F_3, F_7	**Covered.** Signature-validation behaviour is directly exercised.
R_4	F_2, F_3	**Covered.** Timestamp presence and timestamp-related consistency are exercised.
R_5	F_9, F_{12}, F_{14}	**Covered.** Cross-service consistency is exercised through confirmation-related, duplication-related, and list-alignment anomalies.
R_6	None currently mapped	**Gap.** A dedicated test case would be needed to verify that detected inconsistencies are explicitly and correctly reported in logs or error outputs.
R_7	$F_7, F_9, F_{18}, F_{20}, F_{22}$	**Covered.** Protocol-level signature and certificate-related validity are exercised.
R_8	F_{15}, F_{18}, F_{19}	**Covered.** Inclusion in the valid voter list and list-structure constraints are exercised.
R_9	F_2, F_7, F_{17}	**Covered.** BDOC profile and structural correctness are directly exercised.
R_{10}	F_5	**Covered.** Encrypted-ballot presence and naming logic are exercised through payload absence.
R_{11}	$F_{31}, F_{32}, F_{33}, F_{34}, F_{35}, F_{36}$	**Covered.** Multiple manipulations target last-vote selection logic and its invariants.
R_{12}	F_{31}, F_{32}	**Covered.** Voter-to-ballot linkage is exercised through swap scenarios.
R_{13}	F_{37}	**Covered.** Paper-override enforcement is exercised directly.
R_{14}	None currently mapped	**Gap.** The current corpus does not include a separate fault scenario targeting the requirement that anonymisation occurs only after cleansing has completed.
R_{15}	F_{25}, F_{32}	**Covered.** District-attribute preservation is exercised directly.

Req. ID	Mapped fault scenarios	Coverage status and remarks
R_{16}	None currently mapped	**Gap.** The current corpus does not include a separate fault scenario targeting the separation of cryptograms from signatures during anonymisation.
R_{17}	None currently mapped	**Gap.** A dedicated malformed-output-format stimulus would be needed.
R_{18}	None currently mapped	**Gap.** The current corpus does not include a separate fault scenario targeting the requirement to generate both human-readable and machine-readable lists of e-voters.
R_{19}	None currently mapped	**Gap.** The current corpus does not include a separate fault scenario targeting the requirement to produce activity and error logs.

Fault scenarios not directly mapped to a single explicit requirement

Table 9: Fault scenarios evaluated primarily through system-level invariants rather than a single explicit documented requirement.

Constraint type	Fault scenarios	Remark
Time windows and input limits	F_4, F_8, F_{10}, F_{11}	These expose boundary conditions of size and timing enforcement, but are not all directly tied to a single published requirement in the current list.
Output set integrity	F_{26}, $F_{26\text{-}2}$, F_{27}, F_{28}, F_{29}, F_{30}	These are evaluated primarily through set-consistency and accounting invariants rather than through a single explicit requirement statement.
Replay or uniqueness anomaly	F_{13}	This is treated as a specification or documentation gap in Publication II: the anomaly is plausible, but no explicit public requirement states an anti-replay or cryptogram-uniqueness rule.

Curriculum Vitae

1. Personal data

Name	Tarvo Treier
Date and place of birth	13 November 1982, Võru, Estonia
Nationality	Estonian

2. Contact information

Address	Tallinn University of Technology, School of Information Technologies, Department of Software Science, Akadeemia tee 15a, 12618 Tallinn, Estonia
E-mail	tarvo.treier@taltech.ee

3. Education

2007–…	Tallinn University of Technology, School of Information Technologies, Information and Communication Technology, PhD studies
2005–2007	Tallinn University of Technology, Faculty of Information Technology, Informatics, MSc *cum laude*
2002–2005	Tallinn University of Technology, Faculty of Information Technology, Informatics, BSc

4. Language competence

Estonian	native
English	good

5. Professional employment

2013–…	Tallinn University of Technology, Lecturer
2012–2013	STACC OÜ, Research Fellow
2009–2013	Tallinn University of Technology, Research Fellow
2008–2009	Tallinn University of Technology, Assistant
2022–…	SBX Tools OÜ, Founder and CEO
2007–…	Tautar OÜ, Co-founder and CEO
2007–2012	Ericsson Eesti AS, System Integrator
2006–2007	Hansapank AS, Software Engineer
2004–2006	Uptime OÜ, Software Engineer

6. Defended theses

- 2007, *Application of preprocessing to machine learning based on monotone systems* (in Estonian), MSc, supervisor Prof. Rein Kuusik, Tallinn University of Technology, Faculty of Information Technology, Department of Informatics

7. Scientific work
Papers

1. T. Treier and K. Düüna. Identifying and solving a vulnerability in the Estonian internet voting process: Subverting ballot integrity without detection. *IEEE Access*, 12:197766–197782, 2024

2. T. Treier. Beyond the happy path: A safety-critical audit methodology for Estonia's i-voting processing application. *IEEE Access*, 13:203429–203443, 2025

3. T. Treier. Re-voting under surveillance: National eID transaction logs as a threat to coercion resistance in Estonian internet voting. In D. Duenas-Cid, P. Roenne, M. Volkamer, M. Blom, P. Gaudry, I. Borucki, L. Loeber, and A. Debant, editors, *Electronic Voting*, pages 191–207, Cham, 2025. Springer Nature Switzerland

4. V. Sor, Plumbr OÜ, T. Treier, and S. N. Srirama. Improving statistical approach for memory leak detection using machine learning. In 2013 IEEE International Conference on Software Maintenance, pages 544–547, 2013

5. T. Treier. A new effective approach for solving the rules conflict problem. In 2011 Third Pacific-Asia Conference on Circuits, Communications and System, pages 1–3, 2011

6. R. Kuusik, T. Treier, G. Lind, and P. Roosmann. Machine learning task as a diclique extracting task. In 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, volume 1, pages 555–560, 2009

7. P. Roosmann, L. Võhandu, R. Kuusik, T. Treier, and G. Lind. A new inductive learning algorithm based on monotone system theory. In Proceedings of the 8th Conference on Applied Computer Scince, ACS'08, page 310–316, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS)

8. P. Roosmann, L. Võhandu, R. Kuusik, T. Treier, and G. Lind. Monotone systems approach in inductive learning. International Journal of Applied Mathematics and Informatics, 2(2):47–56, 2008

Invited talks

1. T. Treier. *How transparent is Internet voting technology?* (in Estonian), Cybersecurity Commission of the Estonian Academy of Sciences conference *Trust and Trustworthiness 2025: The Future of an Independent Digital State*: 22 September 2025, Tallinn, Estonia.

Elulookirjeldus

1. Isikuandmed

Nimi	Tarvo Treier
Sünniaeg ja -koht	13.11.1982, Võru, Eesti
Kodakondsus	Eesti

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Tarkvarateaduse instituut, Akadeemia tee 15a, 12618 Tallinn, Estonia
E-post	tarvo.treier@taltech.ee

3. Haridus

2007–…	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Info- ja kommunikatsioonitehnoloogia, doktoriõpe
2005–2007	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatika, MSc *cum laude*
2002–2005	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatika, BSc

4. Keelteoskus

eesti keel	emakeel
inglise keel	hea

5. Teenistuskäik

2013–…	Tallinna Tehnikaülikool, lektor
2012–2013	STACC OÜ, teadur
2009–2013	Tallinna Tehnikaülikool, teadur
2008–2009	Tallinna Tehnikaülikool, assistent
2022–…	SBX Tools OÜ, asutaja ja tegevjuht
2007–…	Tautar OÜ, kaasasutaja ja tegevjuht
2007–2012	Ericsson Eesti AS, süsteemiintegraator
2006–2007	Hansapank AS, programmeerija
2004–2006	Uptime OÜ, programmeerija

6. Kaitstud lõputööd

- 2007, *Eelkorrastuse rakendamine monotoonsetel süsteemidel baseeruvale masinõppele*, MSc, juhendaja Prof. Rein Kuusik, Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatikainstituut

7. Teadustegevus

Teadusartiklite, konverentsiteeside ja kutsutud ettekannete loetelu on esitatud ingliskeelses elulookirjelduse osas.

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-528-2 (PDF)