A framework for comparing the security of voting schemes

Atte Juvonen

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, October 1, 2019

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author

Atte Juvonen

Työn nimi — Arbetets titel — Title

A framework for comparing the security of voting schemes

Oppiaine — Läroämne — Subject

Computer Science

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year October 1, 2019	Sivumäärä — Sidoantal — Number of pages 131

Tiivistelmä — Referat — Abstract

We present a new framework to evaluate the security of voting schemes. We utilize the framework to compare a wide range of voting schemes, including practical schemes in real-world use and academic schemes with interesting theoretical properties. In the end we present our results in a neat comparison table.

We strive to be unambiguous: we specify our threat model, assumptions and scope, we give definitions to the terms that we use, we explain every conclusion that we draw, and we make an effort to describe complex ideas in as simple terms as possible.

We attempt to consolidate all important security properties from literature into a coherent framework. These properties are intended to curtail vote-buying and coercion, promote verifiability and dispute resolution, and prevent denial-of-service attacks. Our framework may be considered novel in that trust assumptions are an *output* of the framework, not an *input*. This means that our framework answers questions such as ”how many authorities have to collude in order to violate ballot secrecy in the Finnish paper voting scheme?”

ACM Computing Classification Systems (CCS):

Computers in other domains → Computing in government → Voting / election technologies
Security and privacy → Formal methods and theory of security → Security requirements
Security and privacy → Security services → Privacy-preserving protocols

Avainsanat — Nyckelord — Keywords

voting, elections, systems, protocols, security

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**
1.1 Different types of voting schemes 2
1.2 Failure modes 3
1.3 Trust 4
1.4 Verifiability 4
1.5 Disposition of this thesis 7

2 Research approach **8**
2.1 Research questions 8
2.2 Prior work 8
2.3 Contributions 9
2.4 Scope limitations 10
2.4.1 Exotic ballots 11
2.4.2 Soundness of cryptographic building blocks 11
2.4.3 Man-in-the-middle attacks 12
2.4.4 Cost analyses 12
2.5 Research methods 13

3 Comparison framework **13**
3.1 Goals 14
3.2 Threat model 14
3.3 Assumptions 15
3.3.1 A line in the sand 15
3.3.2 Adversaries are computationally bounded 16
3.3.3 Voters are unable to produce unique ballots 16
3.3.4 Voter registration is not vulnerable to impersonation . 17
3.4 Summary of consolidated properties 18
3.5 Confidentiality 21
3.5.1 Ballot secrecy 22
3.5.2 Receipt-freeness 22
3.5.3 Coercion-resistance 24
3.5.4 Fairness 25
3.6 Integrity 26
3.6.1 Individual verifiability 26
3.6.2 Universal verifiability 32
3.6.3 Eligibility verifiability 33
3.6.4 Dispute resolution 34
3.7 Availability 36
3.7.1 Denial-of-service resistance 36

4 Building blocks of voting schemes **38**
4.1 Code voting 38
4.2 Public Bulletin Board 39
4.3 Randomized encryption 40
4.4 Re-encryption 41
4.5 Threshold cryptosystem 41
4.6 Plaintext Equivalence Test 42
4.7 Zero-knowledge proofs 42
4.8 Fiat-Shamir technique 43
4.9 Designated verifier proofs 44
4.10 Homomorphic encryption 45
4.11 Mix networks 46
4.12 Randomized Partial Checking 49

5 Case reviews of voting schemes **50**
5.1 In-person paper voting in Finland 51
5.2 In-person paper voting with Floating Receipts 55
5.3 In-person paper voting with Prêt à Voter 61
5.4 Remote paper voting in Switzerland 67
5.5 Remote e-voting in Switzerland 69
5.6 Remote e-voting in Australia 74
5.7 Remote e-voting in Estonia 77
5.8 Remote e-voting with Helios 80
5.9 Remote e-voting with Civitas 84

6 Comparison **101**
6.1 Comparison table 102
6.2 Guidance for interpreting results 103
6.3 Key takeaways from the comparison 105
6.4 Future work 107

Acknowledgements **107**

References **108**

A Opinionated advice for policymakers **118**

B Opinionated thoughts on trust, verifiability and understandability **119**

C Blockchain and P2P voting schemes **121**

D Reviews of similar prior work **125**

> *An election is coming. Universal peace is declared, and the foxes have a sincere interest in prolonging the lives of the poultry.*
>
> – George Eliot

1 Introduction

In recent years, politicians in many countries have been pushing towards digitalization of voting methods, which has sparked security concerns from both the general public and cryptography experts. Most democratic countries have a long history with traditional paper voting and it is largely trusted by people. In contrast, electronic voting methods are largely seen as a ”black box”.

Some concerns are related to integrity. Perhaps the greatest fear with electronic voting is that a foreign power is able to hack the election and manipulate the results without anybody noticing. It is not enough to prevent manipulation and produce the correct outcome: in order to have a stable democracy, elections must also convince people that the outcome is correct – even the losers of the election must be convinced. [8]

Some concerns are related to confidentiality. For example, if a voter can vote from their home, then a spouse might coerce the voter to vote for a specific candidate [20]. As another example, some electronic voting methods offer voters receipts which prove how they voted [2]. This could lead to large-scale vote-buying and coercion campaigns. Voters might be offered money to vote a certain way or they might be coerced with threats of violence or shaming. In fact, confidentiality of elections is considered to be so important that the right to secret elections was declared as a fundamental human right in U.N.'s declaration of human rights [98].

If we only cared about integrity, designing a voting scheme would be easy: we could simply publish a list of how everyone voted. Then everyone could verify that their vote appears on the list and that the sum of all votes is correct. However, lack of confidentiality would lead to aforementioned vote-buying and coercion problems. These conflicting requirements between integrity and confidentiality make voting such a hard problem. The good news is that there is a rich variety of voting schemes (both paper and electronic kind) and some of them attempt to provide the best of both worlds: confidentiality and evidence for the integrity of the election.

In this thesis we present a framework for comparing the security of voting schemes. We utilize this framework to compare a wide range of different schemes, including practical schemes in real-world use and academic schemes with interesting theoretical properties.

1.1 Different types of voting schemes

When trying to classify voting schemes, we identify 2 main axes of interest:

1. Where – Is voting confined to a supervised environment?
2. How – Is the vote casted on a physical or virtual ballot?

With regards to axis 1, we define *in-person voting* as the kind where voting is allowed only in specific, supervised environments like the polling booth. Likewise, we define *remote voting* as the kind where voting is allowed outside supervised environments. This distinction is important because voting outside supervised environments is naturally more susceptible to vote-buying and coercion [20].

With regards to axis 2, we define *paper voting* as the kind where the primary mechanism for casting and tallying votes is done on paper ballots. Likewise, we define *e-voting* as the kind where the primary mechanism for casting and tallying votes is done on electronic machines. We exclude telephone voting schemes from our scope. Note that many schemes incorporate both elements. For example, an e-voting scheme may produce a paper trail for auditing, or a paper vote scheme may include electronic elements to assist in filling out the ballot. In any case, we classify voting schemes in the following 4 categories according to our axes of interest:

In-person paper voting	In-person e-voting
Remote paper voting	Remote e-voting

Many readers will probably associate voting schemes with governmental elections. Voting schemes are also used for referendums (voting on decisions as opposed to nominating people). Although we sometimes use election-related terminology, everything in this thesis applies to referendums equally well.

Voting schemes are also commonly used by many non-governmental entities, such as corporations and student unions. In literature these use cases are often referred to as *low-coercive* environments [2], because the risks for coercion and vote-buying are perceived to be much smaller compared to governmental elections and referendums (at least in part because the stakes are perceived to be lower). In many cases[1] these organizations settle for remote e-voting.

Several countries have initiated small-scale experiments with remote e-voting and at least a dozen countries currently offer remote e-voting in *low-impact*[2] settings. However, remote e-voting currently has high impact in only three countries: Estonia, Switzerland and Australia.[3]

1.2 Failure modes

Voting schemes can succumb to both intentional and accidental failures. In section 3 we classify these failures according to the CIA triad of confidentiality, integrity and availability. Roughly speaking, confidentiality is about protecting the secrecy of the votes, integrity is about preventing manipulation of the tally, and availability is about successfully running an election from beginning to end.

While we would prefer to make voting schemes robust against all kinds of failures, it may not be feasible due to the conflicting requirements. A common strategy in the design of voting schemes, then, is to shift failure modes from more severe to less severe. For example, the introduction of *verifiability* does not prevent manipulation of the tally; it merely reveals when it happens. This essentially shifts the failure mode from a catastrophic integrity failure (such as a foreign power entirely fabricating the tally without anyone noticing) into a less severe availability failure (such as forcing a new election, which is of course bad, but orders of magnitude less severe than stealing the election). In addition, by decreasing the potential benefit for attacking, the incentives are changed, hopefully resulting in less attacks overall.

[1] Examples: Debian, ACM, IEEE, HYY.

[2] As an example of low-impact settings, many countries limit access to remote e-voting to absent voters (such as citizens living abroad or military personnel).

[3] This claim is based on our good-faith effort to research remote e-voting in different countries. It represents our opinion regarding what is impactful and what is not. We address these three countries in detail in section 5.

We discuss these failures in terms of *which participants need to misbehave* in order for a failure to occur. A well-designed voting scheme will not suffer catastrophic failures due to a few misbehaving participants, even if they are privileged authorities.

1.3 Trust

Trust is the central, overarching theme in voting schemes. However, in this context it has two distinct meanings, which can give rise to confusion [73]. Social scientists often talk about *increasing* the public's trust towards the voting system, whereas computer scientists often talk about *reducing* trust – in particular, reducing the need to blindly trust authorities when they declare an election outcome. We prefer elections to be based on evidence rather than blind trust. So whenever we mention ”trust” in this thesis, we refer to the latter meaning (computer science perspective). The only exception is appendix B, where we discuss trust in the former meaning (social science perspective).

1.4 Verifiability

Verifiability is another recurring concept in this thesis with two distinct approaches. Although they are not mutually exclusive, we are going to focus on only one of them. Next we will describe the approach that we are going to ignore (verification of equipment), and after that we are going to describe the approach that we are going to focus on (verification of outcome).

Verification of equipment

The traditional approach to verifying a voting system is to verify that all of its parts are operating correctly [79][93][8]. Whether those parts are computers or punchcard machines, the idea is the same: government representatives sit down to decide exactly how they want equipment to operate. These rules are written down and a costly certification program is created to stamp equipment as government-certified. The certification program might be augmented by public reviews of source code and similar methods.

These government certification programs stifle competition and technological progress without providing tangible security benefits [79][93][56]. At times they even harm security. For example, a voting machine running Windows XP was certified as secure in the U.S., but security updates were not installed (presumably due to cost issues of certifying the updates), so an outdated version of Windows XP – with WiFi turned on – kept running on the voting machine as of 2014 [69].

As another government-certified horror story, the source code of a DRE machine was accidentally leaked by its manufacturer (DRE means direct-recording electronic voting machine, a device that a voter uses to input their vote in supervised conditions, usually without leaving a paper audit trail). The manufacturer, Diebold (currently operating as *Premier Election Solutions*), provides DRE machines for numerous high-stakes elections in the U.S. The leaked source code was analyzed by Kohno et al. [49] and the analysis revealed a complete disregard for security. The leaked system does not provide even rudimentary security measures. For example, voting data is transmitted from polling places to central tabulation unencrypted and without authentication. This could potentially allow anyone with basic programming skills to manipulate election results without detection.

Even if manufacturers of voting machines applied industry standard best practices like timely installation of security updates, encrypting data in transit, and so forth, it is highly unlikely that they would be able to write code which works as intended. A rogue employee might intentionally hide a vulnerability in code, or an honest employee might accidentally create a vulnerability that remains undetected. *The Underhanded C Contest*[4], a long-running competition for hiding vulnerabilities in plain sight, illustrates how difficult it can be to verify that a piece of code functions as auditors believe it functions. A similar competition was arranged specifically for hiding vulnerabilities in DRE voting applications [4]. Furthermore, even if the source code of a DRE system is published and somehow verified to function as intended, there are no guarantees that the code running inside actual DRE machines is the same as the one published.

Equipment can fail even without intentional sabotage. Stark and Wagner [93] detail numerous examples of electronic voting machines losing votes and failing to count votes correctly. In one case a voting machine actually reported negative votes for a candidate.

In summary, we should not rely on verification of equipment. Certain processes (such as public review of source code) can certainly augment security, but we should not fool ourselves into thinking that we can verify how a computer operates.

[4] http://www.underhanded-c.org/ (accessed on 20.9.2019)

Verification of outcome

We don't *really* care how the computers within a voting system are operating – we care insofar as to gain confidence in the outcome. But we can actually design voting schemes in such a way that we can verify the outcome directly without verifying the equipment. The high-level idea is that we have inputs going into a system and outputs coming out of the system. Instead of attempting to verify what the system does, we can simply verify that the outputs are correctly generated from the inputs.

A simple example to illustrate this idea is the "public" voting scheme we described earlier (where everyone's votes are public). In this example we still need a computer or a human to count the votes and publish the official tally, but the voters can verify that the tally is correct by comparing the inputs and outputs of the system, instead of reviewing source code for the "official" vote-counting computer.

Schemes which have good verifiability properties are often referred to as E2E ("end-to-end") verifiable.[5] We will present actual examples of verifiable voting schemes in section 5. They are often fairly complicated to defend against various attacks, but the core of these schemes often follows the same pattern as the "public" scheme verifiability example:

1. The voter gets some kind of receipt of their vote. They can use this receipt to confirm that their vote has been recorded. (In some schemes the receipt can not be used to prove to other people how they voted, so it does not promote vote-buying and coercion.)

2. Anyone can verify that the tally is properly counted from recorded votes. (Often the votes are published in encrypted form and cryptography is used to verify that the tally of votes is correct.)

The cryptographic verification procedures described in this thesis may sound overwhelming at times. It is good to remember that normal people would not be expected to understand these procedures; they can (mostly) click a button and have a computer do all the work for them. Naturally, this has some implications for the understandability of the voting system. We argue in appendix B that this isn't a problem.

A concept closely related to verifiability is *software independence* [79]. A scheme is said to be software independent when an error in its software

[5] We avoid using the "E2E" term since it misleadingly implies actual verifiability from end to end, but is commonly used to describe schemes which do not fully provide such verifiability. For example, Civitas is commonly referred to as E2E verifiable, and as we will show in section 5, it does not provide end to end verifiability. Our interpretation is that authors use this term loosely to describe schemes which have merely good verifiability properties.

can not cause an undetected change in election outcome. This may be achieved with cryptographic verification [20] or it may be achieved by augmenting the electronic record with a voter-verifiable paper audit trail [61] (provided that the paper trail is actually audited [93]).

A more general concept – covering both paper and e-voting schemes – is *evidence-based elections* [93]. A paper voting scheme can be designed to produce cryptographic [17] or non-cryptographic [80] evidence for the correctness of the outcome. Even a traditional paper voting scheme – which is not designed to produce such evidence – can be augmented with compliance audits and risk-limiting audits in order to make its outcome verifiable [8].

The high-level idea behind a risk-limiting audit is that random ballots are examined until we can be confident that the election outcome is correct. Before the audit begins we set a limit – for example, 99% – and the auditors continue examining random ballots until the sample that indicates higher than 99% probability that the election outcome is correct. This means that in a close election auditors have to review more ballots than we would in a landslide election. Risk-limiting audits are surprisingly cost-effective: in a typical election only a tiny portion of ballots have to be audited until we know the outcome to be correct with a very high probability. If the risk-limiting audit fails to convince us that the outcome is correct, a full manual recount is triggered. Risk-limiting audits are convincing only if auditors can verify that the ballots themselves have not been tampered with. For this we need compliance audits, which can produce evidence that the audit trail is sufficient. [58][10][93]

1.5 Disposition of this thesis

The remainder of this thesis is divided into sections as follows:

- Section 2 describes our research.
- Section 3 describes our framework for comparing voting schemes.
- Section 4 describes cryptographic building blocks used in voting schemes.
- Section 5 contains case reviews of voting schemes (from our research perspective described in section 2, utilizing the framework provided in section 3, assuming that the reader is familiar with the building blocks described in section 4).
- Section 6 summarizes our findings in a comparison table.

Our main contributions are the framework and the comparison. Appendices A and B contain additional discussion which is on the fringes of our scope. Appendices C and D contain justifications for some claims and decisions made during this research.

2 Research approach

In this section we articulate our research questions, compare this thesis to similar prior work, highlight our contributions, define our scope and describe our research methods.

2.1 Research questions

We want to investigate the security of voting schemes from a practical standpoint (under realistic assumptions). We want to provide a level ("apples-to-apples") comparison to illustrate differences between different schemes. We articulate the following research questions:

RQ 1. *What strengths and weaknesses do different schemes have relative to each other?*

RQ 2. *Are some of the weaknesses a result of unavoidable tradeoffs?*

RQ 3. *Which voting schemes are most suitable for different use cases?*

2.2 Prior work

Several authors [8][64][78][60][100][62][29][57][85] list security properties discovered in literature. Some of them [64][78] do not attempt to consolidate overlapping and conflicting properties into a coherent framework. Many of them [78][60][100] do not provide a comprehensive comparison of voting schemes. Some works [8][62][29][57] have other issues. One article [85] provides both a great framework and a great comparison – however, it was written in 2004, so it does not compare modern schemes. We elaborate on these claims in appendix D.

In summary, we were unable to find a thorough, apples-to-apples comparison on the security of modern voting schemes. We hope that this thesis fills that void.

2.3 Contributions

The main contributions of this thesis are the following:

- We provide a framework for comparing the security of voting schemes (section 3). Our framework may be considered novel in that trust assumptions are an output of the framework, not an input. The collection of properties in our framework is also novel – a result of deconstructing and consolidating existing properties from literature in a creative way in order to satisfy a specific set of goals.
- We apply the framework in case reviews of several real-world and academic schemes (section 5). Unlike prior work in literature[6], we evaluate all schemes under the same assumptions in an attempt to provide an apples-to-apples comparison.
- We compact the results of the case reviews into a simple comparison table (section 6). In stark contrast to prior work in literature, we justify every single claim presented in the comparison table.[7]

In addition, we present some minor contributions:

- We articulate the notion of *displaced votes* (sections 3, 5 and 6). This is a serious threat in several voting schemes and it has been severely overlooked in prior literature.
- We articulate a more stringent requirement for dispute resolution (than what is typically described in literature). We argue why this is necessary. (Section 3.6.4.)
- We identify a missing element of the original Floating Receipts -scheme (bar code, section 5.2).
- We identify a cast-as-intended vulnerability in Floating Receipts and propose an amendment to fix the vulnerability (section 5.2).

[6] Comparisons in literature typically evaluate different schemes under different assumptions (whichever assumptions the scheme's authors used). Even when schemes are largely evaluated under the same assumptions, authors often evade hard questions by marking properties as ”conditionally accepted” (implying additional assumptions, often without specifying what they are). Needless to say, we feel that this approach makes it difficult to compare schemes to each other. We elaborate on these claims in appendix D.

[7] Readers who are wondering how we came to draw a particular conclusion can simply click on the name of the scheme in the comparison table. Comparison tables in prior work are typically unjustified – values in the tables seemingly appear from thin air. We elaborate on this claim in appendix D.

- We propose an amendment to Floating Receipts to improve the amount of receipts which would be verified in a real-world scenario (section 5.2).
- We identify a denial-of-service vulnerability in the audit procedures of Helios (section 5.8).
- We provide the first comprehensive analysis of individual verifiability in Civitas (section 5.9).
- We identify a forced abstention vulnerability in Civitas' smart card extension (section 5.9).
- We propose an amendment to Civitas' smart card extension to improve its coercion-resistance (unrelated to the forced abstention vulnerability) (section 5.9).

2.4 Scope limitations

This literature review compares security of different voting schemes. Aspects other than security, such as usability or accessibility, are not considered. We want to highlight that our focus is specifically on voting schemes; not their corresponding implementation, best practices of software development, or other related subjects. A *voting scheme* refers to the high-level protocol description; the theoretical "rules" according to which voting and tallying is organized. A *voting system* can be thought of as the implementation of a voting scheme.[8] In practice, voting systems often contain multiple voting schemes to accommodate different voters. For example, Estonia's mixed system includes a remote e-voting scheme and an in-person paper voting scheme.[9]

In addition to our intended scope, we had to constrain our workload with further scope limitations, which we describe next.

[8] These concepts are often co-mingled in literature. However, we needed some terms to describe what this thesis is and isn't about, and we chose to use these terms.

[9] In Estonia's system, a voter may override their remote vote by re-voting in-person. Thus, the in-person and remote voting schemes are not clearly separated and one might argue they should be viewed as a conjoined scheme. We consider them separate.

2.4.1 Exotic ballots

The scope of this thesis is limited to simple ballots[10] where the voter selects one choice out of pre-determined options. More exotic ballots may have multiple races, allow a voter to select multiple options for a single race (approval voting) or the expression of the voter may be more nuanced (rank voting, range voting).

We provide the following justification for this scope limitation: Voting schemes can be often modified to support different kinds of ballots without significantly weakening their security properties.[11] This means that a comparison of voting schemes for simple ballots can be useful even if the goal is to select a voting scheme for a more exotic ballot type.

2.4.2 Soundness of cryptographic building blocks

Many voting schemes utilize cryptography. Although we do evaluate the soundness of voting schemes (and in many cases we refute claims made by their authors), we do not evaluate the soundness of cryptographic building blocks utilized in these schemes. The cryptographic building blocks described in section 4 are ”standard” in cryptography (as in, they are not novel inventions created for voting schemes).

We do not imply that all of the underlying cryptography is secure. In fact, the security of cryptography relies extensively on unproven assumptions[12] and the history of cryptography is littered with broken constructions which were once thought to be secure[13]. This is all very interesting, but outside the scope of this thesis.

[10] We coined the term *simple ballots* because the terms available in literature conflate ballot type with voting system type. For example, *Plurality voting* is the closest term we found in literature. It conflates ballot type (”choose one”) with method of determining winners (”single winner based on who receives most votes”). We are concerned with ballot type only; not with how the results of the tally are utilized. For example, in Finnish parliamentary elections there are multiple winners and they are not determined solely based on individual vote counts. If we had scoped to plurality voting, we would have unnecessarily excluded many voting systems, including the Finnish parliamentary elections.

[11] Rivest and Smith [80] provide great examples on how a voting scheme can be extended to support multiple ballot types.

[12] Many cryptographic proofs are reductions to certain hardness assumptions, such as the Decisional Diffie-Hellman assumption [9]. In other words, we may not be able to prove that breaking something is hard, but we may be able to prove it is *at least as hard* as breaking something else: a well-known problem which no-one has been able to break.

[13] The history of hash functions is a good example of breakage:
https://valerieaurora.org/hash.html (accessed 25.7.2019)

2.4.3 Man-in-the-middle attacks

Man-in-the-middle attacks refer to attacks which require privileged network access, such as a colluding Internet Service Provider or WiFi hotspot. Although they are a serious threat in many other contexts, we consider them to be an academic curiosity in the context of voting schemes (we would like to emphasize again that our research concerns voting schemes, not their corresponding implementations[14]). Standard cryptographic methods[15] can be used to protect votes from eavesdropping and tampering in transit. The remaining threat is the identification of *who voted*. While some authors [5] consider this information harmless, it can in theory be used to launch forced abstention attacks [46] (a form of coercion and vote-buying). However, the Tor anonymization network [77] can be used to adequately defend from this threat in practice.[16] We do not attempt to conclusively prove these claims; we are merely justifying our decision to exclude man-in-the-middle attacks from our scope.

2.4.4 Cost analyses

The cost of arranging elections is of great practical importance, but it is outside the scope of this thesis, as we focus on voting schemes, not implementations. However, in order for this thesis to have any practical relevance, we do consider costs in one aspect: we disregard absurdly expensive schemes. To summarize, we *evaluate voting schemes which are believed to have feasible costs, but we do not differentiate how costly these schemes are relative to each other.*

Furthermore, we do not consider cost analyses on mounting attacks (such as cost estimation for discovering and weaponizing vulnerabilities, or cost estimation for brute-force or denial-of-service attacks). We reviewed a few articles of this nature, but in our view, they were heavily opinionated and relied too much on dubious assumptions. In other words, our justification for leaving these cost analyses outside our scope is that we were unable to find quality research articles on this subject.

[14] Man-in-the-middle vulnerabilities have been discovered [35][14] in real-world *implementations* of voting schemes.

[15] Mainly referring to encryption, authentication, and Public Key Infrastructure (certificate authorities in particular).

[16] We are aware of de-anonymization attacks targeted at Tor users. We find it incredibly unlikely that these attacks would ever be mounted just to reveal who voted (they can not be used to reveal what was on the ballot).

2.5 Research methods

This thesis is a literature review. We began our literature search by making Google and Google Scholar searches related to voting security in general. We soon expanded our search to specific voting schemes and specific security properties. As we discovered relevant works, we often followed their citations backwards to discover related articles. Sometimes we followed citations forward – for example, if we wanted to verify that a proposed construction was not later broken.

We selected articles for further reading based on title, abstract, publish date and citations. After partially reading articles we made further eliminations. For example, many older research papers describe obscure voting schemes which rely heavily on trusted authorities. These schemes seem inferior compared to more modern schemes, so we did not see merit in analyzing them further. We made exceptions cases where the system is actually still used and has real-world impact. For example, any system which is currently in use by a large portion of a country for government elections deserves attention. To summarize our selection process: we were interested in voting schemes which were either *good* or *significantly used in practice*.

The bulk of the work was spent crafting the comparison framework. We spent endless hours tweaking our scope, assumptions and properties. Every time we learned something new about voting schemes, we recognized aspects of the framework that were lacking. It turns out that classifying things is hard.

3 Comparison framework

Every author in voting literature has a different perspective and different focus. It is difficult to compare different systems without any common ground. In order to find common ground, we make the following contributions:

1. We articulate a set of goals for a comparison framework.
2. We present a framework to fulfill these goals as well as possible.

To the best of our knowledge we are the first to articulate a set of goals for a comparison framework in this field. Although many frameworks have been proposed, the authors have not explained which goals they hope to achieve with their frameworks.[17]

[17] For more discussion on other frameworks, we refer to appendix D.

3.1 Goals

These are the goals we wish to fulfill with our framework:

1. The framework must be useful in *differentiating* different schemes. (We want to highlight the strengths and weaknesses of schemes relative to each other, and the properties we choose for comparison should help us in this task.)

2. Results of the comparison should be *unambiguous*. (If two people read the same results, they should not walk away with different conclusions.)

3. Results of the comparison should be useful to *non-experts*. (Presentation of results should be simplified to the point where laypersons can mostly understand them. For example, informal definitions are easier to understand than formal definitions.)

4. Properties should be defined *without unnecessary overlap*. (Literature is filled with terms which refer to almost identical concepts. We do not want to merely repeat everything we found. Similar concepts should be either consolidated together or deconstructed into clearly separate terms.)

5. Properties should be defined so that *everything important is covered*.

6. Assumptions should be *practical*.

7. Assumptions should *remain constant* throughout the comparison (instead of using one set of assumptions to evaluate one scheme and a different set of assumptions to evaluate a different scheme, as is usually done in prior literature. We elaborate on this in appendix D.).

8. *Familiarity*: properties should leverage established definitions and terms as much as possible (introducing new concepts and terms only if absolutely necessary).

Next we will present a framework which attempts to fulfill these goals.

3.2 Threat model

The typical approach in literature is to define a single adversary with a wide range of capabilities (along with some crucial, limitating assumptions), and then analyze how vulnerable a particular voting scheme is against that particular adversary. This type of analysis often yields very interesting results, but we do not think it is realistic or useful.

In the real world there are many adversaries with different capabilities; Inguva et al. [41] provide an excellent overview. We have to make certain tradeoffs in the design of our voting schemes to account for these adversaries and it is important to acknowledge these tradeoffs instead of hiding them under unrealistic assumptions of a singular adversary. In addition, it is difficult to compare different systems when each author is defending against a different adversary.

We acknowledge that multiple adversaries exist and they have different capabilities. Therefore, *we do not make overarching trust assumptions regarding which participants are honest.* Instead, for each type of property, we describe the weakest possible trust assumptions under which the property is secured. For example, instead of saying ”ballot secrecy is guaranteed” [given our trust assumptions], we might say ”ballot secrecy is guaranteed as long as at least one authority is honest”. This allows us to illustrate how different trust assumptions are required to secure different properties within a scheme. Likewise, it allows us to illustrate how different trust assumptions are required to secure the same property within different schemes. In other words, trust assumptions are an output of our framework, not an input.

We attempt to consolidate all important security properties from literature. We present our properties in relation to terminology that is prevalent in voting literature. Next we will present our assumptions and after that we will present the security properties.

3.3 Assumptions

In order to provide an apples-to-apples comparison we need to have unified assumptions (as opposed to evaluating each scheme under a different set of assumptions). Note that *trust assumptions* are an output of our framework, not an input, which is why we do not set trust assumptions in this section.

3.3.1 A line in the sand

Anyone who has ever been to a magic show knows how difficult it can be to detect deception, even for an observer who is perfectly alert and waiting for something to happen. Now consider the process of counting thousands of physical ballots: many people sit in a room, shuffling through paper for hours on end. If observers are unable to spot magic tricks at a magic show, what hope do they have of spotting magic tricks during this long and arduous process of vote counting? Some ballots might disappear or end up in the wrong pile without anyone noticing. In fact, we already know that manual vote counting often produces incorrect results even when everyone in the room is honestly trying their best [32].

The physical world offers endless opportunities for deception. People could hide cameras in voting booths to violate ballot secrecy[18]. People could replace pens in voting booths with disappearing ink pens[19]. The digital world offers opportunities for deception as well. Someone could infect a vote-counting machine with malware to manipulate the count. We could design a voting scheme to allow anyone to verify the count, but what if someone infects all computers with the same malware so there isn't a clean computer to verify with?

We could throw our hands in the air and say "anything is possible", but that would not lead to a very useful analysis. Instead, we draw a line in the sand and say that some threats are realistic, other threats are not. We do not think hiding cameras in voting booths is a realistic threat, but we do think people are going to photograph their own ballots [40] if they have an incentive to do so. We do not think disappearing ink pens are a realistic threat, but we do think people are willing to steal entire ballot boxes [26]. We expect everyone to have differing opinions where the line should be drawn. That's fine. We still have to draw the line somewhere.

3.3.2 Adversaries are computationally bounded

Some cryptographic schemes are secure against any adversary, including adversaries with unlimited ("unbounded") computational power [63]. Other schemes are secure only against computationally bounded adversaries. On the face of it, this does not appear to be a practical distinction. After all, all adversaries in the real world are computationally bounded. We decided to reduce our workload by consciously ignoring this aspect of voting schemes.

3.3.3 Voters are unable to produce unique ballots

If the voting scheme publishes all ballots, it is crucial that a voter can not produce unique (or likely unique) ballots. Otherwise a potential coercer or vote-buyer may identify voters from published ballots by

[18] On rare occasions cameras have been hidden in voting booths: https://www.timesofisrael.com/operation-moral-standards-inside-likuds-election-day-arab-surveillance-program/ (accessed on 21.9.2019)

[19] On rare occasions disappearing ink pens have been used to sabotage elections: https://www.thetimes.co.uk/article/votes-in-invisible-ink-just-vanish-in-ballot (accessed on 21.9.2019)

coercing them into filling ballots in a specific way. For example, *write-in votes* enable voters to nominate new choices on the ballot (such as a person who is not officially running in the race). A coercer might demand that the voter inserts a specific, arbitrary string as their write-in vote (to prove that the voter has wasted their vote). Unique ballots can be produced in other ways as well. For example, if a ballot contains too many races, the combination of candidates on a ballot may be unique.

In order to simplify later analysis, we assume that voters will be unable to produce unique ballots (or ballots which are likely to be unique). In our opinion this assumption is realistic in typical single-race ballots.[20,21]

3.3.4 Voter registration is not vulnerable to impersonation

Voter registration is needed to ensure that only authorized persons can vote and no voter votes more than the allowed number of times (”one person one vote”). Some countries, such as the United States, have an explicit voter registration process, whereas other countries, such as Finland, provide automatic voter registration for eligible persons. In any case, we want to make a distinction between ”identification credentials” and ”voting credentials”. In many voting schemes, the voter uses their identification credentials to acquire voting credentials during the registration procedure. We assume that voters can not be impersonated during this registration procedure.

We acknowledge that this assumption is somewhat problematic. In reality, vote-buyers may co-operate with voters to impersonate them during the registration phase. This circumvents any later protections against vote-buying and coercion; if a vote-buyer is able to acquire legitimate voter credentials, they will be indistinguishable from a legitimate voter. A vote-buyer who acquires voter credentials directly from the registrar can also be confident in the authenticity of the credentials, and thus avoid being cheated by the vote-seller, making vote-buying economically viable.

Clarkson et al. [20] propose various practical defenses against this impersonation threat. One of their recommendations is to require supervised conditions for registration. Another recommendation is to make identification credentials so valuable that voters will be reluctant to sell them. As an example they mention the Estonian ID card,

[20] If there are multiple races, especially if an exotic ballot type is used, voters may be able to produce unique ballots. A practical mitigation in these instances is to separate each race onto different ballots, as recommended by Rivest and Smith [80].

[21] One common case where our unique ballot assumption falls short is handwritten ballots. Handwriting is inherently compatible with steganography, allowing voters to encode additional information on their ballots. However, we consider this threat vector to be more ”academic” than ”practical”.

which can be used – in addition to acquiring voter credentials – to sign economic transactions, such as bank loans. Naturally, most users would be reluctant to hand such power over to vote-buying entities.

We provide the following justification for assuming that voter registration is not vulnerable to impersonation: in order to provide coercion-resistance there must be *some kind of time window* during which the voter is not controlled by the coercer. If the voter is controlled by the coercer at all times, it will be impossible to avoid coercion. In the context of remote voting schemes, the natural choice for this trusted time window is registration, because it can be arranged under supervised conditions (unlike remote voting itself).

3.4 Summary of consolidated properties

In this section we summarize the properties in our framework to provide the reader with the "big picture" before delving into the details. The order of properties is thematic and is not related to importance. We encourage readers to peek into the comparison table (section 6.1) before reading further. This should provide some intuition regarding how the following properties are utilized in the comparison.

P1. *Malware on voting device is unable to violate ballot secrecy.* If the voting device is a computer or similar, the voter must be able to obfuscate their choice with code voting.

P2. *Malware on voting device is unable to manipulate votes.* If the voting device is a computer or similar, voters must be convinced of two things. First, that their personal vote has not been manipulated by malware (or bugs) on the voting device. Second, that a large-scale malware campaign is not manipulating votes en masse. One way to prevent these possibilities entirely is by using a code voting scheme. Another way is by allowing voters to verify (via secondary device) that their votes have been cast as intended and recorded as cast, with a possibility to re-vote if errors are discovered. However, since only a small portion of voters typically use verification procedures, it is not sufficient to rectify only the discovered errors. In the "verify-and-revote" solution, we require that voters be able to prove the existence of a large-scale malware campaign in order to allow courts to interfere with the election before larger damage is done. Furthermore, the "verify-and-revote" solution must not be susceptible to clash attacks.

P3. *Voter is able to keep their ballot as secret.* (To clarify, the voting scheme does not leak information which could substantially help an adversary to guess how a voter voted, given that the adversary already has access to the final tally.)

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* We define large-scale vote-buyer as an adversary who does not possess the ability to physically accompany voters, but does possess the ability automate any computational workflow. For example, a large-scale vote-buyer can automate verification of receipts (if such a thing is possible) or they can electronically vote on behalf of voters (if such a thing is possible). (To clarify, certain forms of re-voting can be used to defraud vote-buyers and thus satisfy this property.)

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* This covers two attacks: forced-abstention attack (proof of not voting) and randomization attack (proof of voting a random candidate). Both of these attacks intend to prevent a voter from exercising their right to vote. Large-scale vote-buyer is defined in P4.

P6. *Voter is unable to prove to their spouse how they voted.* Spouse is representative of adversaries with the ability to physically accompany voters in some parts of the electoral process, but without the ability to collude with corrupted insiders. (The adversary can not accompany voters during registration or inside a voting booth, and the adversary can not accompany the voter during the entire time window of the voting process). (To clarify, certain forms of re-voting may fulfill this propery.)

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* This covers two attacks: forced-abstention attack (proof of not voting) and randomization attack (proof of voting a random candidate). Both of these attacks intend to prevent a voter from exercising their right to vote. Spouse is defined in P6.

P8. *Voter can ensure their ballot is not accidentally spoiled.* More precisely, if the ballot is accidentally spoiled by the voter's actions, the voter has an ability to detect this and re-vote. (To clarify a corner case, this definition also includes spoiling the ballot by entering incorrect credentials in schemes like Civitas.)

P9. *Voter can ensure their vote is recorded as cast.* The voter has an ability to verify that their vote has been recorded as cast (with no susceptibility to clash attacks). If the voter receives negative confirmation, no confirmation at all, or discovers a discrepancy between how their vote was casted versus how it was recorded, they can re-vote. Note that we expect this verification to be mandatory (otherwise there is a risk that a large campaign will manipulate many votes and only the verifying portion of voters have their votes recorded properly). The voter may physically observe their ballot falling in a box or the voter may rely on a trusted voting device to show confirmation of digital receipt (corrupted voting devices are considered separately in P2). (To clarify, we accept any reasonable dispute resolution, even if the voter needs help of the election officials in order to re-vote.)

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Even if voters can ensure that their vote is recorded, an authority may delete their vote later. An additional threat is present in some schemes: an adversary (such as the voter's spouse) may replace their vote by re-voting with their credentials. Another variation of this threat is present in some schemes where only the first vote counts. In that case, the adversary may pre-empt the voter's vote by voting before them. (We do not demand dispute resolution for this property, because it would be inherently impossible to provide for the ”replaced or pre-empted” conditions.)

P11. *The tally is counted correctly from recorded votes.* In addition, we require adequate dispute resolution in case of discrepancies in order to satisfy this property.

P12. *No ballot stuffing.* All votes which affect the final tally correspond to a real voter, no voter corresponds to more than one vote, and malformed votes are not counted. Furthermore, fraudulent votes can not be added to voters who did not vote. Voters who did not vote will be extremely unlikely to take initiative in verifying their non-vote, so we do not accept verification mechanisms which rely on the initiative of these non-voters. In addition, we require dispute resolution to satisfy this property. In other words, if ballot stuffing is detected, it must be rectified. Note that it is not sufficient to remove only ”the detected subset” of fraudulent votes from the tally; either all fraudulent votes must be detected and removed or the election results must be invalidated and a new election must be organized.

P13. *Denial-of-service resistance.* Absence of any known attacks which could be undertaken to deny availability to voting or tallying. Attacks by authorities are included. General DDoS attacks are excluded.

Additional clarifications:

- Privileged insiders, such as poll workers, talliers and voting system vendors, are grouped under the umbrella of *authorities*. A group of insiders working together, or working for the same employer, is considered to represent a single authority.
- We consider malware (and bugs) on voting devices as a separate concern, in properties P1 and P2. All other properties reflect cases where software on voting devices is working as intended. Note that we make no assumptions regarding software on *other* devices (for example, software on a vote-counting computer).
- If a scheme is software independent, we do not consider an exclusive vendor at all problematic. If a scheme is not software independent and it has an exclusive vendor, then we consider that vendor to be a single point of failure for a variety of things, including undetected vote manipulation. In some cases this causes us to flag a property with ”Holds as long as no authorities are misbehaving”.
- We assume the presence of malicious voters and malicious outsiders. Furthermore, we assume that a large number of malicious voters are willing to collude with a corrupted authority (in cases where an authority is corrupted).

Next we delve into the details to motivate why these properties are important and how they relate to familiar terms in voting literature. We omit justifications for why we consolidated the properties in exactly this fashion – it was a long process of constant tweaking motivated by the goals in section 3.1. We organized the following analysis according to the CIA triad: Confidentiality, Integrity and Availability.

3.5 Confidentiality

In this section we present confidentiality properties organized according to familiar concepts from voting literature: ballot secrecy, receipt-freeness, coercion-resistance and fairness. The first three are about protecting *individual* voter’s information, the fourth is about protecting *aggregate* information.

3.5.1 Ballot secrecy

We define *ballot secrecy* as the ability of the voter to keep their votes as secret.

More precisely, the voting scheme does not intentionally publicize or unintentionally leak information which could substantially improve an adversary's guess regarding how a voter voted (given that the adversary already knows the final vote counts for each candidate). This precise definition covers some important corner cases.

Suppose that 100% of voters vote for the same candidate. It is trivial for an adversary to find out how each voter voted simply using the final vote counts for each candidate. However, the adversary's guess is not improved by any additional information leaked by the voting scheme. Therefore, using this definition, ballot secrecy is not compromised in this example.

In some cases an adversary is able to make educated guesses about voters' choices, even though the guesses are not 100% accurate. For example, correlation attacks on ThreeBallot fall inside this category [96]. Since our research questions concern practical viability (rather than theoretical guarantees), we decided to draw a line in the sand and declare *substantial* leaks of information as violating ballot secrecy (this mainly affects the VAV scheme in our comparison). As such, our definition is most influenced by Juels et al. [45] and Strauss [96].

The following properties in our comparison are related to ballot secrecy:

P1. *Malware on voting device is unable to violate ballot secrecy.* If the voting device is a computer or similar, the voter must be able to obfuscate their choice with code voting.

P3. *Voter is able to keep their ballot as secret.* (To clarify, the voting scheme does not leak information which could substantially help an adversary to guess how a voter voted, given that the adversary already has access to the final tally.)

3.5.2 Receipt-freeness

We define *receipt-freeness* as the inability of the voter to prove to a static adversary how they voted.

Our definition is in line with Juels et al. [45], who provide a brief history of receipt-freeness in literature. According to them, the term appeared first in [6] by Benaloh and Tuinstra, although the concept was independently introduced by Niemi and Renvall in [68].

Receipt-freeness can be seen as a stronger form of ballot secrecy. Whereas ballot secrecy prevents the *authorities* from violating confidentiality, receipt-freeness prevents the *voter* from violating confidentiality. We can also look at it from the perspective of who is protected: ballot secrecy protects the voter from coercion, whereas receipt-freeness protects the general public from vote-buying. Although, naturally, receipt-freeness also provides stronger protection against coercion. For example, ballot secrecy is enough to prevent mild forms of coercion, such as your co-workers giving you disapproving looks after you voted the wrong way, but is not enough to prevent stronger forms of coercion, such as a local thug demanding a receipt that proves you voted the right way.

When voters are unable to prove how they voted, vote-buyers risk being cheated out of their money. This hopefully[22] has the effect of turning large-scale vote-buying into a fruitless pursuit.

The following property in our comparison is related to receipt-freeness:

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* We define large-scale vote-buyer as an adversary who does not possess the ability to physically accompany voters, but does possess the ability automate any computational workflow. For example, a large-scale vote-buyer can automate verification of receipts (if such a thing is possible) or they can electronically vote on behalf of voters (if such a thing is possible). (To clarify, certain forms of re-voting can be used to defraud vote-buyers and thus satisfy this property.) (This property also includes simulation attacks.)

[22] In some cases receipt-freeness is not enough to prevent large-scale vote-buying. For example, cultural and economic conditions in Argentina paved the way for large-scale vote-buying [12]. Argentinian political parties provided food, clothing, and other necessities to people in exchange for their votes. Voters often ”returned the favor” despite that they could theoretically cheat.

3.5.3 Coercion-resistance

We define *coercion-resistance* as the inability of the voter to prove to an interactive adversary anything about their participation in the voting process. Paraphrasing from [45] and [20], coercion-resistance covers receipt-freeness and the following four additional attacks:

- Interactive adversary: the voter is unable to convince an adversary who is physically present during the voting process (such as a coercive spouse in a remote e-voting scheme).
- Simulation attack: the voter is unable to convince an adversary even if they loan their voting credentials to them.
- Randomization attack: the voter is unable to convince an adversary that they voted for a random candidate.[23]
- Forced-abstention attack: the voter is unable to convince an adversary that they voted at all.

The terminology around receipt-freeness and coercion-resistance can be misleading. It may give a false impression that receipt-freeness is entirely about preventing vote-buying whereas coercion-resistance is entirely about preventing coercion. This is not the case. Both receipt-freeness and coercion-resistance protect against both coercion and vote-buying. Coercion-resistance offers more protection against both threats. In fact, these threats are almost identical from a game theoretic perspective[24].

Providing coercion-resistance in a practical setting is an ambitious goal, as illustrated in [20] by Clarkson et al.: *"In remote voting, the coercer could even be the voter's employer or domestic partner, physically present with the voter and controlling the entire process. Against such coercers, it is necessary to ensure that voters can appear to comply with any behavior demanded of them."* In fact, it is so ambitious that no voting scheme achieves it without setting unrealistic trust assumptions (or sacrificing verifiability).

Furthermore, some attacks (such as the simulation attack) are clearly more serious than others (such as the forced-abstention attack).[25] Due

[23] It may not be immediately obvious how a randomization attack may be possible if the voter can not prove who they voted for. Hirt and Sako [39] provide an example in the context of remote e-voting. For an example in the context of in-person paper voting, see section 5.3.

[24] If we exclude mild forms of coercion and some special cases, we can expect a vote-seller to make the same decisions as a coerced voter. In fact, many articles in literature use these terms synonymously.

[25] For example, Bell et al. [5] describe publication of who voted as "harmless".

to these reasons, we spent a lot of time considering different options of how we should bundle these attacks in the properties of our comparison. This is the end result:

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* We define large-scale vote-buyer as an adversary who does not possess the ability to physically accompany voters, but does possess the ability automate any computational workflow. For example, a large-scale vote-buyer can automate verification of receipts (if such a thing is possible) or they can electronically vote on behalf of voters (if such a thing is possible). (To clarify, certain forms of re-voting can be used to defraud vote-buyers and thus satisfy this property.)

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* This covers two attacks: forced-abstention attack (proof of not voting) and randomization attack (proof of voting a random candidate). Both of these attacks intend to prevent a voter from exercising their right to vote. Large-scale vote-buyer is defined in P4.

P6. *Voter is unable to prove to their spouse how they voted.* Spouse is representative of adversaries with the ability to physically accompany voters in some parts of the electoral process, but without the ability to collude with corrupted insiders. (The adversary can not accompany voters during registration or inside a voting booth, and the adversary can not accompany the voter during the entire time window of the voting process). (To clarify, certain forms of re-voting may fulfill this propery.)

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* This covers two attacks: forced-abstention attack (proof of not voting) and randomization attack (proof of voting a random candidate). Both of these attacks intend to prevent a voter from exercising their right to vote. Spouse is defined in P6.

3.5.4 Fairness

Releasing intermediate results of an election could provide upcoming voters with an advantage over those who have already voted. We define *fairness* as all voters having access to substantially the same information.

Note that most definitions of fairness are stricter than ours. For example, Rjaskova [81] and Fouard et al. [29] have opted for a definition which excludes participants from having ”any knowledge” about the partial tally before the end of the election. In our opinion this definition is unnecessarily strict because any in-person voting scheme leaks a negligible amount of information to voters around voting areas. For example, if you see your neighbor walk to the voting booth, and you know they favor a particular candidate, you do gain some knowledge about the partial tally before the end of the election – but the knowledge you gain is not substantial.

In earlier drafts of this thesis we had a fairness property in the comparison. However, we noticed that for all of the voting schemes in our comparison, the conditions necessary to violate fairness were always the same as the conditions required to violate ballot secrecy. In other words, the fairness property was redundant. This may not always be the case for all voting schemes, but it is the case for all schemes in our comparison. Furthermore, we do not consider the release of intermediate results to be a realistic threat to voting systems in practice. Due to these reasons we decided to exclude the fairness property from our comparison.

3.6 Integrity

The majority of articles related to integrity in voting schemes focus on *verifiability*: the detection of errors. In many cases it is unclear what should be done when errors are detected. This aspect of voting schemes is referred to as *dispute resolution*.

3.6.1 Individual verifiability

We define *individual verifiability* as the ability of the voter to convince themselves that their vote was counted appropriately. We deconstruct individual verifiability into 3 consecutive phases: cast as intended, recorded as cast, and count as recorded.

Note that this is a contentious term. Almost every author uses a one-sentence description similar to ours, but as they deconstruct this high-level description into low-level details, they end up with a wide variety of different definitions. Our definition consolidates all of the important aspects from various definitions offered in literature. We provide this simple justification for our definition: *every low-level detail that we include is necessary to satisfy the agreed-upon high-level description.* As we describe the 3 phases of individual verifiability, we provide examples to illustrate how each phase is necessary.

Some readers may wonder why a single ”count as intended” verification at the end of the voting process would not be sufficient. The answer is that voters have no recourse to correct mistakes at the end of the voting process (except, perhaps in the case where a large proportion of voters report mistakes). It is preferrable to have all three of these verifications so that some errors can be corrected in time. In addition, voting schemes often provide some of these verifications, but not all of them, so this deconstruction helps to differentiate voting schemes.

Cast as intended

We define *cast as intended verification* as the ability of the voter to verify that their intent was interpreted correctly. If their vote was not interpreted correctly, the voter has the opportunity to fix the error before depositing their vote in the ballot box (physical or otherwise).

For example, imagine a voting scheme where a voter goes to a polling place, selects their preferred candidate on a computer, and the computer prints out a filled ballot. The voter can now verify that their intent was interpreted correctly before depositing the unambiguous ballot into the ballot box. If the selection was incorrect or there is a problem with the computer, the problem can be corrected before any damage is done.

As an example of a voting scheme which doesn't provide ”cast as intended” verifiability, consider a typical paper voting scheme, such as the one used in Finland. A voter goes to the polling place, writes down handwritten symbols representing their preferred candidate, and deposits the ballot *before* their intent is interpreted. In every Finnish election a significant amount of ballots are disqualified due to ambiguous markings.[26] Figure 1 provides an example of an ambiguous vote.

Some ballots are also disqualified due to markings which may identify the voter, leading to potential vote-buying. Further exacerbating these issues is the fact that election officials have a lot of leeway in choosing which ballots to disqualify, opening the door for potential biases to influence the count[27]. These issues could be avoided entirely by using a voting scheme which provides ”cast as intended” verifiability.

Every known input method for casting the voter's intention on a ballot suffers from non-negligible amount of failures in practice. As an example of another input method, consider the use of optical scan machines. Many jurisdictions use them to interpret ballots where each candidate has a circle next to their name and the voter is expected

[26] No data is available on the amount of ballots disqualified due to ambiguous markings. However, disqualified ballots overall typically represent less than 1% of ballots in major elections in Finland (according to data by Ministry of Justice). Ambiguous ballots are some fraction of this, because many disqualified ballots are intentionally spoiled.

[27] In one instance the validity of hundreds of votes was contested:
https://www.hs.fi/politiikka/art-2000005170716.html (accessed 27.5.2019)

to color their chosen candidate's circle. Voters often smudge the area around the circle, causing failures during automatic counts, which sometimes leads to contentious manual recounts.

Punchcard systems represent another common input method, where a machine is used to punch holes in ballots (similar to optical scan machines, except a machine is used to physically mark the circle). One might imagine that the machine eliminates smudging errors, and it does, but at the cost of introducing another error: sometimes the punched area is not entirely cut from the ballot, leading to ambiguity again. Additionally, both optical scan systems and punchcard systems also suffer from voters accidentally selecting too many options. These issues are often referred to in literature as *undervoting* and *overvoting*. [64]

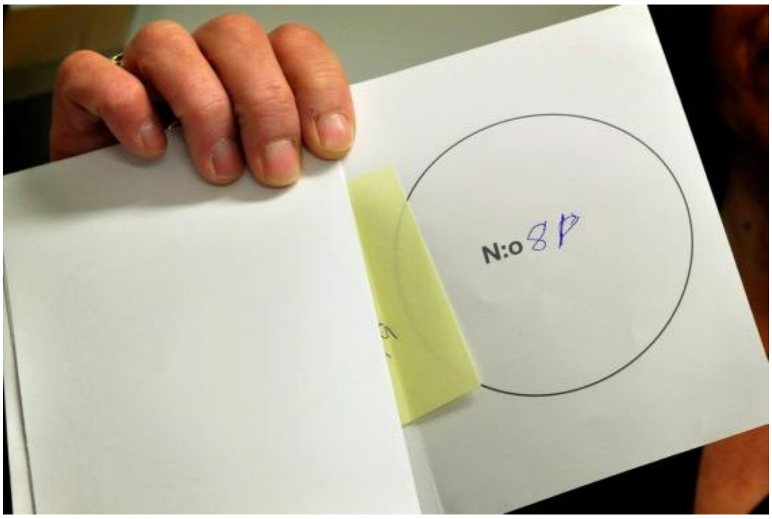

Figure 1: Disqualified vote from 2012 Finnish Municipality Elections. Possibly a "80" which was later corrected to "81". This image serves as an example of problems arising in voting schemes which do not provide "cast as intended" verification. Photograph by Else Kyhälä, republished with permission.

Currently available literature suggests that the only way to avoid cast-as-intended failures is to provide cast-as-intended verifiability. Implementing "cast as intended" verifiability can also backfire in unexpected ways. Finland experimented with in-person e-voting in 2008 Municipality elections (in 3 municipalities). Roughly 2% of votes were not counted[28], apparently because the e-voting machine in use presented the user with an "are you sure?" type pop-up after the user attempted to vote, and some voters walked out without noticing this

[28] https://www.kho.fi/fi/index/ajankohtaista/tiedotteet/2009/04/9.4.2009-khomaarasikunnallisvaalituusittavaksikolmessakunnassa.html (accessed 25.9.2019)

pop-up. Publically available documentation does not clarify why this pop-up existed, but it is likely that the pop-up was added for ”cast as intended” verifiability: so that users who misclicked had a chance to change their vote. This case also illustrates the perils of adding any type of user-facing complexity to a voting scheme, no matter how small.

The following properties (P2 and P8) are related to cast-as-intended. In P2 the cause of the error is the voting device. In P8 the cause of the error is the voter.

P2. *Malware on voting device is unable to manipulate votes.* If the voting device is a computer or similar, voters must be convinced of two things. First, that their personal vote has not been manipulated by malware (or bugs) on the voting device. Second, that a large-scale malware campaign is not manipulating votes en masse. One way to prevent these possibilities entirely is by using a code voting scheme. Another way is by allowing voters to verify (via secondary device) that their votes have been cast as intended and recorded as cast, with a possibility to re-vote if errors are discovered. However, since only a small portion of voters typically use verification procedures, it is not sufficient to rectify only the discovered errors. In the ”verify-and-revote” solution, we require that voters be able to prove the existence of a large-scale malware campaign in order to allow courts to interfere with the election before larger damage is done. Furthermore, the ”verify-and-revote” solution must not be susceptible to clash attacks.

P8. *Voter can ensure their ballot is not accidentally spoiled.* More precisely, if the ballot is accidentally spoiled by the voter’s actions, the voter has an ability to detect this and re-vote. (To clarify a corner case, this definition also includes spoiling the ballot by entering incorrect credentials in schemes like Civitas.)

Recorded as cast

We define *recorded as cast verifiability* as the ability of the voter to verify that their vote has been recorded without alterations. It is important to confirm that the ballot has been recorded at the time of voting, so that problems can be corrected before the tallying phase. Network problems, such as denial-of-service attacks, can be potential reasons for failure.

We[29] consider clash attacks [53] to violate recorded-as-cast verifiability. In a clash attack, an adversary shows the same verification to multiple people who voted the same candidate. For example, if only 3 people vote for a candidate, the adversary might change 2 of the votes and avoid detection by showing all 3 voters the same verification (corresponding to the only real vote). A clash attack may be executed by malware on the voting device or by the authority recording votes.

Properties P2 and P9 are related to recorded-as-cast. In P2 the cause of the error is the voting device. In P9 the cause of the error is misbehaving authorities.

P2. *Malware on voting device is unable to manipulate votes.* If the voting device is a computer or similar, voters must be convinced of two things. First, that their personal vote has not been manipulated by malware (or bugs) on the voting device. Second, that a large-scale malware campaign is not manipulating votes en masse. One way to prevent these possibilities entirely is by using a code voting scheme. Another way is by allowing voters to verify (via secondary device) that their votes have been cast as intended and recorded as cast, with a possibility to re-vote if errors are discovered. However, since only a small portion of voters typically use verification procedures, it is not sufficient to rectify only the discovered errors. In the ”verify-and-revote” solution, we require that voters be able to prove the existence of a large-scale malware campaign in order to allow courts to interfere with the election before larger damage is done. Furthermore, the ”verify-and-revote” solution must not be susceptible to clash attacks.

[29] Some authors [50] consider clash attacks to be excluded from individual verifiability, but this would be in direct conflict with the informal definitions for individual verifiability and recorded-as-cast verifiability. If the voter can not verify that their vote has been recorded, then clearly, individual verifiability has not been provided.

P9. *Voter can ensure their vote is recorded as cast.* The voter has an ability to verify that their vote has been recorded as cast (with no susceptibility to clash attacks). If the voter receives negative confirmation, no confirmation at all, or discovers a discrepancy between how their vote was casted versus how it was recorded, they can re-vote. Note that we expect this verification to be mandatory (otherwise there is a risk that a large campaign will manipulate many votes and only the verifying portion of voters have their votes recorded properly). The voter may physically observe their ballot falling in a box or the voter may rely on a trusted voting device to show confirmation of digital receipt (corrupted voting devices are considered separately in P2). (To clarify, we accept any reasonable dispute resolution, even if the voter needs help of the election officials in order to re-vote.)

Count as recorded

We define *count as record verifiability* as the ability of the voter to verify that their vote has been counted in the final tally.

At first glance this verification may seem redundant if we already have recorded-as-cast and universal verifiability. If the voter is able to verify that their vote is part of the recorded collection of votes (recorded-as-cast) and that the final tally of votes is correctly computed from the recorded collection of votes (universal verifiability), it is not immediately obvious why we need to separately verify that the voter's personal vote was count as recorded. However, there are cases where the voter's personal vote may not be counted despite these verifications. If the voting scheme allows multiple votes to be recorded with the same voting credentials, and only counts one of these votes, then an adversary can potentially displace the voter's intended vote with another one (if they somehow gain access to voting credentials). This is why count-as-recorded verification is necessary. Some authors, such as Neumann and Volkamer in [67], have fallen into this trap (see section 5.9 for more details).

Note that this is not a theoretical threat: even if the voting scheme and its corresponding implementation are perfect, the responsibility for handling voting credentials typically falls on the voter. Given how insecurely average voters handle their passwords[30], we have no

[30] According to a 2016 study by Pew Research Center, only 3% of users use a password manager as their primary method to remember passwords. A staggering 65% rely primarily on their memory (indicating high password reuse) while 18% primarily write their passwords down. https://www.pewinternet.org/2017/01/26/2-password-management-and-mobile-security/ (accessed 15.8.2019)

reason to believe that they would handle their voting credentials any better. Therefore, it is reasonable to expect a certain amount of voting credentials to be stolen in practice. In addition, voting credentials may be stolen by a corrupt registrar or reconstructed by adversaries due to faults in the underlying cryptography or its implementation. Individual verifiability – and especially count-as-recorded verifiability – is a way to reveal when these things happen, and correspondingly, increase confidence in the votes which were legitimately counted.

Individual verifiability may seem contradictory to the goal of receipt-freeness: how can a voter gain a proof which is convincing to *themselves*, and yet unconvincing to *others*, even if the voter shares all his secret key material with vote-buyers? Surprisingly, some voting schemes are able to provide both features at the same time. For example, Civitas uses designated-verifier proofs to provide these features (excluding cast-as-intended). We refer to section 5.9 for more details.

The following properties in our comparison are related to count-as-recorded:

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Even if voters can ensure that their vote is recorded, an authority may delete their vote later. An additional threat is present in some schemes: an adversary (such as the voter's spouse) may replace their vote by re-voting with their credentials. Another variation of this threat is present in some schemes where only the first vote counts. In that case, the adversary may pre-empt the voter's vote by voting before them. (We do not demand dispute resolution for this property, because it would be inherently impossible to provide for the ”replaced or pre-empted” conditions.)

P11. *The tally is counted correctly from recorded votes.* In addition, we require adequate dispute resolution in case of discrepancies in order to satisfy this property.

3.6.2 Universal verifiability

We define *universal verifiability* as the ability of anyone to verify that the reported tally is correctly calculated from a published set of (usually encrypted) votes. This definition is not controversial [29], although some authors [74] do use the term to mean something else.

Universal verifiability and individual verifiability together make up the features we referred to in section 1 when we discussed the general idea behind verifiability: voters individually verify that their vote is in a published set of (usually encrypted) votes, and anyone can verify that the tally is correctly computed from this published set of votes. We would like to remind readers that – although these features are incredibly powerful – they are not sufficient. Some threats, such as ballot stuffing, are still unaccounted for. (Ballot stuffing is discussed in 3.6.3 and various vulnerabilities are discussed in detail in section 5.)

We consider universal verifiability within the following property:

P11. *The tally is counted correctly from recorded votes.* In addition, we require adequate dispute resolution in case of discrepancies in order to satisfy this property.

3.6.3 Eligibility verifiability

We define *eligibility verifiability* as the ability of anyone to verify that only legitimate votes were counted. In particular, the following threats must be accounted for:

- Votes can not be malformed (such as a vote which adds a negative number to a candidate's count).
- No votes from non-eligible persons.
- At most one vote per person must affect the final tally (note that in some systems a person may vote multiple times, but only their first/last vote should count).
- Votes can not be undetectably added to voters who did not vote.

We consider aspects of eligibility verifiability within the following property:

P12. *No ballot stuffing.* All votes which affect the final tally correspond to a real voter, no voter corresponds to more than one vote, and malformed votes are not counted. Furthermore, fraudulent votes can not be added to voters who did not vote. Voters who did not vote will be extremely unlikely to take initiative in verifying their non-vote, so we do not accept verification mechanisms which rely on the initiative of these non-voters. In addition, we require dispute resolution to satisfy this property. In other words, if ballot stuffing is detected, it must be rectified. Note that it is not sufficient to remove only "the detected subset" of fraudulent votes from the tally; either all fraudulent votes must be detected and removed or the election results must be invalidated and a new election must be organized.

3.6.4 Dispute resolution

We deconstruct dispute resolution into three aspects:

1. Nonrepudiation: when the voter detects an error, they can prove that the error has occurred.
2. Accountability: the misbehaving authority can be identified.
3. Error correction: the voting scheme includes a process to correct the error.

Furthermore, the voter should be able to prove misbehavior without sacrificing ballot secrecy. Our definition is largely in line with the description in [47].

For example, suppose the voter can verify that their vote is recorded, but it is not counted in the final tally due to misbehavior in a mix network. In particular circumstances it may be possible for the voter to prove an error, but it may be unclear which authority in the mix network is misbehaving. The error correction process may involve the court system.

Nonrepudiation and accountability provide the following benefits: [60]

1. They pave the way towards error correction when misbehavior occurs (and subsequently, deter authorities from misbehaving).
2. They discredit fraudulent claims of authority misbehavior (and subsequently, deter discontent voters from making fraudulent claims).

One aspect of error correction that we were expecting to find in literature – but didn't – is how to proceed with error correction if there are signs of widespread manipulation. Some voting schemes imply that only the discovered instances of manipulation should be corrected [16]. That is insufficient.

Suppose a voting scheme allows a verifying voter to detect manipulation with 100% certainty (this is not always the case [16]). Suppose an adversary corrupts 5% of untrusted voting machines with malware. Suppose 10% of voters verify their votes.[31] This means that 5% of votes will be manipulated, but only 0.5% of votes will be detect as manipulated. If election officials or courts simply rectify these discovered manipulated votes, then 90% of manipulated votes still end up in the tally, representing 4.5% of total votes.

[31] Empirical evidence suggests that even fewer voters bother with verification procedures [8].

This example illustrates why it is insufficient to simply rectify the *discovered* instances of manipulation. When manipulation has been detected, *all* instances of manipulation must be rectified. This is not an easy problem. One approach to solve this is to make verification mandatory. For example, code voting systems require the voter perform certain verification steps. Another approach is to halt the election and arrange a new election (after investigating how votes were manipulated and somehow preventing it from happening again). We fear that in practice authorities would be reluctant to do this. We have already seen Estonian officials shrug at potential signs of vote manipulation and continue with the election regardless [91]. We have also seen Australian officials sweep large scale verification failures under the carpet.[32]

We decided to bundle dispute resolution properties together with their corresponding verifiability properties. These properties in our comparison include dispute resolution:

P2. *Malware on voting device is unable to manipulate votes.* If the voting device is a computer or similar, voters must be convinced of two things. First, that their personal vote has not been manipulated by malware (or bugs) on the voting device. Second, that a large-scale malware campaign is not manipulating votes en masse. One way to prevent these possibilities entirely is by using a code voting scheme. Another way is by allowing voters to verify (via secondary device) that their votes have been cast as intended and recorded as cast, with a possibility to re-vote if errors are discovered. However, since only a small portion of voters typically use verification procedures, it is not sufficient to rectify only the discovered errors. In the ”verify-and-revote” solution, we require that voters be able to prove the existence of a large-scale malware campaign in order to allow courts to interfere with the election before larger damage is done. Furthermore, the ”verify-and-revote” solution must not be susceptible to clash attacks.

P8. *Voter can ensure their ballot is not accidentally spoiled.* More precisely, if the ballot is accidentally spoiled by the voter’s actions, the voter has an ability to detect this and re-vote. (To clarify a corner case, this definition also includes spoiling the ballot by entering incorrect credentials in schemes like Civitas.)

[32] https://pursuit.unimelb.edu.au/articles/where-s-the-proof-internet-voting-is-secure (accessed on 28.9.2019)

P9. *Voter can ensure their vote is recorded as cast.* The voter has an ability to verify that their vote has been recorded as cast (with no susceptibility to clash attacks). If the voter receives negative confirmation, no confirmation at all, or discovers a discrepancy between how their vote was casted versus how it was recorded, they can re-vote. Note that we expect this verification to be mandatory (otherwise there is a risk that a large campaign will manipulate many votes and only the verifying portion of voters have their votes recorded properly). The voter may physically observe their ballot falling in a box or the voter may rely on a trusted voting device to show confirmation of digital receipt (corrupted voting devices are considered separately in P2). (To clarify, we accept any reasonable dispute resolution, even if the voter needs help of the election officials in order to re-vote.)

P11. *The tally is counted correctly from recorded votes.* In addition, we require adequate dispute resolution in case of discrepancies in order to satisfy this property.

P12. *No ballot stuffing.* All votes which affect the final tally correspond to a real voter, no voter corresponds to more than one vote, and malformed votes are not counted. Furthermore, fraudulent votes can not be added to voters who did not vote. Voters who did not vote will be extremely unlikely to take initiative in verifying their non-vote, so we do not accept verification mechanisms which rely on the initiative of these non-voters. In addition, we require dispute resolution to satisfy this property. In other words, if ballot stuffing is detected, it must be rectified. Note that it is not sufficient to remove only ”the detected subset” of fraudulent votes from the tally; either all fraudulent votes must be detected and removed or the election results must be invalidated and a new election must be organized.

3.7 Availability

In order to successfully run an election, voters must be able to register and vote, and votes must be tallied. We consider the availability of registering, voting and tallying to be of great practical importance.

3.7.1 Denial-of-service resistance

We want to make a distinction between low-resource denial-of-service attacks (henceforth DoS) and distributed denial of service attacks (DDoS). Although a practical voting system must be sufficiently protected from

DDoS attacks, they are a general threat to all internet facing services. This threat must be mitigated at network-layer, and as such, it is clearly out of scope for a thesis on voting schemes (which are implemented at application-layer). Furthermore, this threat is currently mitigated in practice by large corporations, such as CloudFlare, which offer DDoS protection to their clients. It is not infeasible to think that a real-life voting system would simply buy DDoS protection from these corporations just as many commercial website operators do.

Although DoS and DDoS attacks are technically attacks on the availability of the system, they can be applied *selectively* to attack the integrity of the vote. For example, if carefully chosen users or regions are denied availability while everyone else is able to vote, the outcome of an election might change. [8]

A voting scheme must be protected from DoS attacks. For example, some voting schemes [20] accept all votes sent by everyone and then expend huge computational resources to validate each vote. A system that is designed like this will essentially grind to a halt when faced with malicious traffic from a single computer, as we will later demonstrate.

Furthermore, a voting scheme that is designed like this will be unable to utilize network layer protection against this malicious traffic. For example, the attacker could be routing their traffic through the Tor network. If all traffic from Tor was simply filtered out, some legitimate voters would be filtered out as well. Clearly it is crucial for any practical voting scheme that the computational cost to defend against an attack can not be disproportionate to the cost of attacking.

In addition, misbehaving authorities may refuse to accept votes or tally. Running an election should be possible despite this. We did not find definitions related to this in literature.[33]

The following property in our comparison covers both application-level DoS threats and authority-related DoS threats:

P13. *Denial-of-service resistance.* Absence of any known attacks which could be undertaken to deny availability to voting or tallying. Attacks by authorities are included. General DDoS attacks are excluded.

[33] The closest definition we found was *k-authority robustness* in [81], which states that the scheme must be robust against k misbehaving authorities and unlimited misbehaving voters. However, our definition does not embed trust assumptions (number of misbehaving authorities is an output in our framework) and our definition is limited to availability (their definition mixes in integrity properties).

4 Building blocks of voting schemes

This section describes the underlying building blocks which are commonly used in voting schemes. We assume the reader is already familiar with the following topics: time complexity, cryptographic hash functions, encryption/decryption, key generation, public-key cryptography, and digital signatures. We encourage readers to look up[34] these concepts if they are not in fresh memory. Readers who do not have a computer science background are encouraged to skip this section entirely. It is possible to think of these building blocks as ”black boxes” which are used for specific purposes without necessarily understanding how these building blocks achieve their purpose.

4.1 Code voting

Remote e-voting typically involves voting on untrusted general-purpose computing devices, such as home computers or smartphones. This leads to the risks we addressed in P1 and P2: malware on the voting device may be able to violate ballot secrecy or even manipulate votes without detection. Code voting refers to various methods which can be utilized to defend against these threats. Next we will provide a general description of these ideas.

Candidate	Vote code	Response code
John	f8h13	kLQ9x
Mike	1WWcZ	orOlk
Sally	MnxB5	yj474

Table 1: Illustration of a code sheet in a code voting scheme.

A voter receives a code sheet in physical mail before the election (illustrated in table 1). When it comes time to vote, the voter opens the voting application on their untrusted device, but instead of inputting their choice in plain text, the voter inputs a corresponding code from the code sheet. Malware on the device can see the code, but because the codes have been randomly assigned and are different for each voter, malware cannot know which candidate is represented by the code, so malware can not violate ballot secrecy. However, malware can still manipulate votes (for example, by pretending to send the vote without actually sending it). This threat can be thwarted by another set of

[34] We recommend Christof Paar’s lecture series on cryptography: https://www.youtube.com/watch?v=2aHkqB2-46k (accessed 22.9.2019)

codes: response codes. Once the vote has been sent, the server responds with a response code, which the voter can compare to the response code on their code sheet. Again, malware has no access to these voter-specific codes, so it will be unable to fake a response code to the voter. [84]

4.2 Public Bulletin Board

Many voting schemes utilize a bulletin board for verifiability purposes. For example, the bulletin board may contain all votes in their encrypted form, the tally from encrypted votes, and a (privacy-preserving) proof that the tally is correctly computed from the encrypted votes. It is crucial that the bulletin board functions as an append-only log. Voters (or other participants) must be able to post legitimate messages on the bulletin board. No message can ever be deleted.

As we discuss voting schemes, we will simply assume that a bulletin board with these properties exists. Although currently available literature does not offer a ”theoretically bullet-proof” construction for a bulletin board, one might argue that various constructions are ”good enough” in practice. Next we will discuss various ways of constructing a bulletin board.

A bulletin board can be hosted on a traditional, untrusted web server. Honesty against deletions can be enforced by having the bulletin board sign messages posted to it. If a message is later deleted, a voter or any outside observer can prove this with the signature. One way to improve the write-availability of a bulletin board in practice is by setting up *relays*. A voter can submit their vote to any subset of relays, which will later post the vote on the bulletin board (as long as one of the relays is honest). This idea was first introduced by Clarkson et al. [20], who called these relays *ballot boxes* (but the idea can be generalized to any data, not just votes). The relay construction can also be leveraged in the case where the bulletin board is corrupted and refuses to accept some legitimate votes (the relays can confirm this on behalf of the voter). Additionally, outside organizations need to verify that the bulletin board does not present different contents to different users.[35]

An alternative implementation choice for bulletin boards is distributed ledgers, such as blockchains. These data structures (typically) provide a trustless, append-only log, which is exactly what we need. Unfortunately, none of the current permissionless distributed ledgers are suitable for large-scale elections [78]. The most established of these

[35] In the context of voting schemes it is typically sufficient for outside organizations to compare their copies of the bulletin board after an election has been completed.

ledgers, Bitcoin and Ethereum, are frequently congested by even modest amounts of traffic[36]. Blockchains are also not designed for the kind of availability guarantees which are required in voting (miners are typically free to reject any entries to the log, such as votes for an unfavorable candidate) [8].

Furthermore, blockchains attempt to solve a much harder problem than what is needed for voting: irreversibility of transactions. With financial transactions it is absolutely crucial that a transaction does not become invalidated later. However, with voting we only require that cheating is detected afterwards; invalidating votes or a tally is fine as long as the errors are corrected before the result becomes official. This is why a trusted server with digital signatures against deletions can provide a sufficient implementation of a public bulletin board [8].

4.3 Randomized encryption

Randomized encryption is a way to prevent brute-forcing when the input space to the encryption function is small or vulnerable to guessing [1]. For example, sometimes a vote only has two options: yes and no. Suppose a voter encrypts plaintext ”yes” using the tallying authorities' public key, resulting in ciphertext ”k3x6”, and sends it to the public bulletin board. An adversary can find out the contents of the vote easily by iterating through all possibilities: first they will try encrypting ”no” and see that results in ciphertext ”qoo9” and next they will try encrypting ”yes” and see that results in ciphertext ”k3x6”, which is a match. If the same input always maps to the same output and the input space is small, it becomes trivial to brute-force encrypted messages.

We can turn a typical encryption scheme into randomized encryption by adding a random value to the input. For example, instead of encrypting ”yes”, we might encrypt ”yes-31Xkd823jd”. If the random values are sufficiently unpredictable (long), then each ”yes” vote yields a different ciphertext and an eavesdropping adversary can no longer recover plaintext values with a naive brute-force strategy.

In practice we will often want to use a more complicated randomized encryption algorithm, because we want to combine it with other useful features, such as re-encryption.

[36] One anecdote to support this claim: https://www.bbc.com/news/technology-42237162 (accessed 25.7.2019)

4.4 Re-encryption

Remember that in a randomized encryption scheme an encrypted vote can have many representations (many ciphertexts which decrypt to the same plaintext). Sometimes, as part of vote anonymization, it is beneficial to change the representation of a vote from one ciphertext to another (without changing the vote itself). A *re-encryption* algorithm [34] allows us to do exactly that. Re-encryption is possible without knowing the plaintext or having the keys to decrypt the ciphertext.[37] Re-encryption is often utilized in mix networks, which we describe in section 4.10.

4.5 Threshold cryptosystem

A *cryptosystem* is a set of three protocols: key generation, encryption and decryption. A *distributed cryptosystem* is a cryptosystem in which a set of entities must cooperate to perform decryption. These definitions are paraphrased from [23]. Typically, we would want to select mutually distrusting entities, such as competing political parties.

A *threshold cryptosystem* is a distributed cryptosystem which provides a k-out-of-n quorum (at least k out of n members need to participate in any computation). Threshold cryptosystems are often used in voting schemes for computing the tally from a collection of encrypted votes without decrypting individual votes (we will return to how this is possible). This construction provides the following advantages:

1. A single misbehaving authority can not violate ballot secrecy (at least k-out-of-n authorities would have to collude to decrypt individual votes).

2. A single misbehaving authority can not prevent the tally from being computed (as long as k-out-of-n authorities are available, computations can be performed).

Threshold cryptosystems were introduced by Desmedt et al. [24] in 1989. Bell et al. [5] describe threshold cryptosystems as a *"straightforward extension of traditional public-key cryptosystems"*. Adida [1] describes how different traditional cryptosystems (El Gamal, RSA and Paillier) can be extended to implement a threshold cryptosystem in voting schemes.

[37] Naturally, the cryptosystem must be constructed in a specific way to facilitate re-encryption.

4.6 Plaintext Equivalence Test

Remember that in a randomized encryption scheme many ciphertexts decrypt to the same plaintext. Sometimes we may want to find out if two ciphertexts decrypt to the same plaintext – without actually decrypting the ciphertext. A plaintext equivalence test [42] can be utilized for this purpose. Next we will provide an example use case to motivate and illustrate the idea.

Suppose we have a randomized encryption scheme where the decryption key is distributed among several authorities. Suppose that voters send their votes accompanied by encrypted credentials. We want to ensure that each credential represents a legitimate voter, but we want to maintain ballot secrecy – we don't want even the talliers (who hold the distributed decryption key) to know which voter sent which vote. The talliers could cooperate to decrypt the credentials, but then they would uncover the identity of the voter. Instead, the talliers can utilize a plaintext equivalence test to compare the encrypted credentials to a list of legitimate (differently encrypted) credentials. This way the talliers uncover only the desired information (that the vote corresponds to *some* legitimate voter), without uncovering information that should be hidden (*which* particular voter it was).[38]

4.7 Zero-knowledge proofs

Suppose we have two parties, a prover and a verifier. The prover wants to convince the verifier that they know a particular secret, but they do not want to reveal the secret itself or any additional information. A construction like this is called a *zero-knowledge proof* [33]. A typical zero-knowledge proof is an interactive protocol where the verifier asks questions from the prover. If the prover knows the secret, they can always deduce the correct answer. Otherwise they can guess. The verifier is convinced by asking so many questions that the probability of guessing all of them correctly is negligible.

Consider this classic example: we have two identical balls, except one of them is red and the other one is blue. The verifier is colorblind and doesn't believe that the prover can differentiate the balls. The prover wants to convince the verifier that they know a secret (which ball is blue, which ball is red), but they do not want the verifier to learn this secret. An interactive protocol for the proof is as follows. The verifier puts the balls behind their back. Then the verifier shows the prover

[38] We omitted many crucial details required for a scheme to be viable. Our intention here was to simply describe the general idea of a plaintext equivalence test in the context of voting schemes. For an actual, complete example, we refer to section 5.9.

one of the balls, and puts it behind their back again. Then the verifier shows another ball and asks if it is the same ball or a different ball. This process is repeated until the verifier is convinced that the prover knows the secret. Crucially, the verifier doesn't learn which ball is red, which ball is blue. They only learn that the prover can differentiate the balls in some fashion.

Zero-knowledge proofs[39] are often leveraged in voting schemes, as we will show in 4.8, 4.9, 4.10 and 4.11.

4.8 Fiat-Shamir technique

Fiat-Shamir technique [27] is a method for collapsing interactive proofs into non-interactive proofs. This is often desirable due to practical problems imposed by interactivity.

For motivation, let us continue with the example that we want the tallying authorities to prove to voters that the tally was correctly calculated. If the proof is interactive, then the authorities need to perform a separate proof together with each voter, imposing both computational overhead and inconvenience. By collapsing the interactive proof into a non-interactive proof, the authorities can instead compute a single proof on their own.

The general idea behind the technique is that the prover simulates inputs of the verifier. This is possible when the actions of the verifier are restricted to producing random inputs (in literature these are often referred to as *Honest-Verifier Zero-Knowledge proofs*, because the verifier is restricted from "dishonestly" choosing adversarial inputs based on responses of the prover [1]).

The difficult part in simulating inputs is how the prover can convince the verifier that the inputs were truly random, and not merely selected to deceive the verifier. The general idea to achieve this is by applying a hash function to prior protocol messages. Output of the hash function will be uniformly random as long as the input has sufficient entropy (unpredictable content by voters). Some additional caveats exist. For example, if the prover can provide part of the input to the hash function, the prover might brute-force until it finds a favorable hash to facilitate cheating. Several approaches exist to mitigate this issue. For example, the prover may use a hash-based commitment scheme [30] to commit

[39] We do not differentiate between zero-knowledge proofs and zero-knowledge arguments [1]. The former is secure against a computationally unbounded adversary, whereas the latter is secure only against a computationally bounded adversary. Since we already assume a computationally bounded adversary, this distinction is not relevant to us and we will try to keep the language simple. So in some cases we will be inaccurately discussing zero-knowledge proofs while actually referring to zero-knowledge arguments.

to a specific random seed at the beginning of the protocol. A different approach is taken by Helios [2], which allows the prover to brute-force inputs for its Fiat-Shamir construction, but has made the search space so large that it would be computationally infeasible for the prover to find a value which enables them to cheat.

4.9 Designated verifier proofs

An ambitious goal in certain voting schemes is to convince the voter – *and only the voter* – that their vote has been counted. One way to fulfill this goal is by using designated verifier proofs, which were first described by Jakobsson et al. in 1996 [44]. Their solution to this seemingly impossible challenge is brilliant in its simplicity.

Suppose that voting authority A wants to prove claim C to voter V, but does not want the proof to be convincing to vote-buyer B. Instead of proving "C is true", A proves "C is true or this proof was created by V". When V receives this proof, she knows that the proof was not created by herself, and therefore C must be true. But if V passes the same proof to B, there is no way for B to know whether C is true or if the proof was created by V in an attempt to deceive B.

Note that designated verifier proofs work even in the scenario where the voter shares all their keying material with the vote-buyer. However, we obviously need to assume that the vote-buyer is not interacting with the voter during the time window when the voting authority is sending the proof (if the vote-buyer can physically observe the proof originating from the authority – as opposed to being faked by the voter – then of course they will be convinced). In literature this requirement is often satisfied by assuming that the registration phase is secure in this aspect [20] [45]. We discuss this assumption further in section 3.3.4. We omit practical implications for now; they are covered within our discussion of Civitas.

In order to gain intuition on how these proofs can be created, let's start with the simplest possible example. Alice wants to convince Bob that Alice has sent a particular message, but does not want Eve to be convinced even if Bob and Eve collude together. In order to achieve this, Alice can encrypt the message to Bob using an encryption key known only to Alice and Bob (for example, using Diffie-Hellman key exchange to generate the shared key and then using AES to communicate with the shared key). When Bob receives the encrypted message from Alice, Bob is convinced that Alice has sent it – after all, no-one other than Alice and Bob know the secret key. However, if Bob passes the key and encrypted message to Eve, Eve cannot know whether the encrypted message was created by Alice or Bob, so Eve is not convinced that it was created by Alice. As Jakobsson et al. note in [44], this method can only convince the receiver *who* sent a particular message; the message itself could still be false. As such, this is not a designated verifier proof. A designated verifier proof would convince the receiver that the message (a proof) is true.

Jakobsson et al. present some examples of actual designated verifier proofs in their article [44]. As a high level example, we will briefly describe the cryptographic building blocks that Jakobsson et al. use to construct a designated verifier proof for an undeniable signature:

1. We begin with an interactive zero-knowledge undeniable signature scheme as described by Chaum in [15]. If the verifier in this scheme records a transcript of the interactive verification, the transcript and key together can be used to convince other people. This undesired property is what we want to eliminate from the scheme. Additionally, we want to transform this interactive scheme into a non-interactive scheme.

2. The scheme involves a commitment stage (where the prover commits to a value with a cryptographic hash function). We modify the commitment stage with a trap-door function tied to the verifier's public key. A trapdoor function provides the plausibility that the verifier could have forged any transcript arising from the communication (or using vocabulary corresponding to the vote-buying example, trapdoor allows the voter to forge a proof for the vote-buyer). Trapdoor commitment schemes are described both intuitively and formally in Fischlin's dissertation [28]. Fischlin demonstrates many examples for the construction of trapdoor functions. Some of them rely on familiar cryptographic elements, such as the Discrete Logarithm problem.

3. We collapse the interactive protocol into a non-interactive protocol by using the Fiat-Shamir technique. Now we have a non-interactive undeniable signature scheme with a designated verifier.

4.10 Homomorphic encryption

Suppose we have a set of encrypted votes. One might imagine that in order to count the votes, we first have to decrypt them. This is one approach (see section 4.11), but it is not the only approach. A form of encryption referred to as *homomorphic encryption* allows anyone – with knowledge of only the public key – to perform certain mathematical operations on ciphertexts. The result of these operations is the same as if the operations had been performed on corresponding plaintexts and then encrypted. For example, suppose we have two ciphertexts: an encryption of 8 and an encryption of 3. Without knowing the plaintexts, and with only the public key, we can compute an encrypted sum of these plaintexts: enc(8) + enc(3) = enc(11). The encrypted sum can be safely decrypted without revealing what the individual inputs were.

The first homomorphic voting scheme was proposed by Benaloh [7]. Next we will describe typical steps in a homomorphic tallying scheme.

Suppose we have a distributed cryptosystem with randomized encryption. In order to facilitate homomorphic tallying, ballots must be constructed in a special way. For example, a ballot might be constructed to contain one ciphertext per each candidate: an encryption of ”1” for the chosen candidate, and an encryption of ”0” for every other candidate. The voter must provide proof that each ciphertext contains either ”0” or ”1” (and not, for example, ”-500”) and proof that at most one of the ciphertexts contains a ”1” (zero-knowledge proofs can be utilized for this purpose).

When the tallying begins, the talliers first compute encrypted sums for each candidate and then jointly decrypt the encrypted sums. Due to the homomorphic property, anyone can verify that the encrypted sums are correct. The talliers still need to prove that decryptions of sums are correct (again, zero-knowledge proofs can be utilized).

4.11 Mix networks

In the context of voting schemes[40], mix networks can be seen as an alternative to homomorphic encryption. Both methods can be used to produce a verifiably correct tally from a collection of encrypted votes, but these methods are fundamentally different: homomorphic encryption is used to compute the tally from encrypted votes *without decrypting individual votes*, whereas a mix network is used to *anonymize votes before decrypting them*. The advantage of using mix networks is that the complete set of ballots is preserved for election auditing. However, this advantage comes at a significant cost: mix network schemes are computationally more demanding, more difficult to design correctly, and more difficult to implement correctly [1].

Suppose we have a collection of encrypted, legitimate votes (as in, we have somehow verified that the set contains only votes from eligible voters, only one vote per voter, and that all the votes are of correct form). A mix network consists of several (mutually distrusting) tallying authorities, who jointly anonymize and decrypt votes. An input

[40] Mix networks were first described by Chaum [18] in 1981. Since then, a vast field of research has grown around them. However, we constrain our focus to the application of mix networks in voting schemes.

collection of encrypted votes is transformed into an output collection of decrypted votes in such a way that no-one – not even the voter or any of the talliers – can tell which vote in the output corresponds to which vote in the input.

Next we will explain two variations of mix networks: decryption mix networks and re-encryption mix networks.

Decryption mix network

Consider a bag of onions. Each onion consists of a vote, protected by three layers of encryption. Votes have been encrypted by voters, each layer with a different public key. The corresponding private keys are held by chefs (talliers) who are tasked with peeling the onions to reveal the votes. However, the chefs are going to peel the onions in a special way in order to preserve secrecy regarding which vote originated from which onion. The first tallier takes the bag of onions and peels (decrypts) the first layer of each onion, and then shakes the bag to shuffle the order of the onions. The second tallier takes the bag and performs the same operations: peels (decrypts) the next layer of each onion, shakes (shuffles) the bag, and passes the bag to the next tallier. This continues until the onions are entirely peeled (decrypted), revealing the actual votes. Figure 2 illustrates this process.

Note that the combination of decryption and shuffling is needed to preserve ballot secrecy. If only shuffling was used, it would be trivial to link inputs to outputs simply by looking for the exact same ballot, since its form doesn't change. If only decryption was used, it would again be trivial to link inputs to outputs, since the 4th input always corresponds to the 4th output. But when decryption and shuffling are performed together, both the form of ballots and the order of ballots changes, making it impossible to link individual inputs to outputs.

In order to prevent corrupt talliers from manipulating the vote, talliers need to prove that they did not alter any votes. We will later explain how, but for now, think of the talliers as performing two tasks: decryption (with proof), and shuffling. Note that the talliers do *not* have to prove the randomness or secrecy of the shuffle. This is because a single honest tallier – a single good shuffle – is enough to maintain the privacy properties of the scheme.

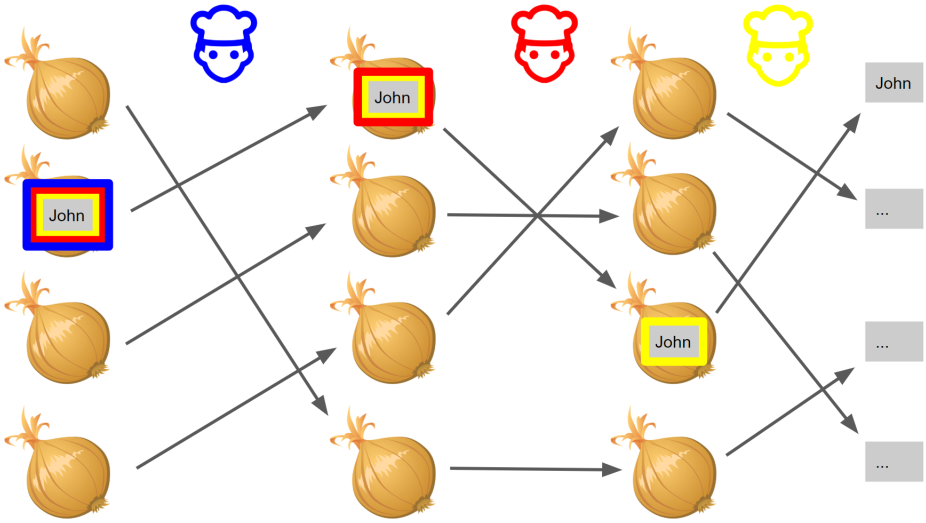

Figure 2: Illustration of a decryption mix network. An onion represents a vote encrypted by several layers of encryption. The illustrated vote was encrypted by the voter using the public keys of the chefs: first with the yellow key, then with the red key, and lastly with the blue key. Decryption must be performed in reverse order. First the blue cook (tallier) peels (decrypts) the blue layer, because the blue cook has the corresponding private key. The blue cook also shuffles the onions. After the red and yellow cooks perform the same steps, plaintext votes are revealed. Ballot secrecy is maintained, because no cook learns which output vote corresponds to which input vote. This illustration is original work with visual assets from freeiconspng.com and icons8.com.

Re-encryption mix network

A re-encryption mix network separates the two tasks into consecutive phases: a mixing phase and a decryption phase. Remember, in order to anonymize votes, each tallier has to change both the order and the form of ballots. In contrast to a decryption mix network, the input collection of votes is encrypted with only a single layer of encryption, so the talliers can no longer change the form of ballots by decrypting layers. Instead, the talliers re-encrypt votes. Again, a proof is needed to convince everyone that no votes were altered. So the mixing phase consists of a chain of talliers, where each tallier mixes the votes with shuffling and re-encryption. After that the mixing phase is complete and the decryption phase begins.

The main advantage of re-encryption mix networks over decryption mix networks is that threshold schemes can be used for decryption, guaranteeing that the tally can be computed even if some talliers fail to co-operate. The use of a threshold also causes some minor vulnerabilities. If more than k-out-of-n talliers collude together, they can decrypt credentials and votes (although they still could not associate credentials to voters – at least not without colluding with the registrar or voters). Note that even if all of the talliers are corrupted, they can not fabricate the tally without being caught (in schemes which have universal verifiability and eligibility verifiability). Another advantage is that the tallying process can be independently re-run without compromising ballot secrecy. [82]

The first practically efficient, verifiable mix networks in voting schemes were described by Neff in [65] and by Furukawa and Sako in [31]. While these schemes use zero-knowledge proofs, other kinds of proofs have also been used in various proposals. Some schemes [17] use a probabilistic method called *randomized partial checking.*

4.12 Randomized Partial Checking

Let us continue with the decryption mix network example illustrated in figure 2. As we noted earlier, cooks in this scheme must somehow prove that they did not manipulate any votes. Otherwise the cooks could simply replace all input votes with freshly generated votes of their preference. Next we will describe how this scheme can be augmented with randomized partial checking in order to produce such proofs.

In figure 2 each cook performed one round of transformations (decryption and shuffling). In the augmented scheme, illustrated in figure 3, each cook will perform two rounds of transformations instead of one. After a cook has committed to particular transformations, it is challenged to reveal a subset of them. A transformation is revealed by proving that a particular output of a transformation is a legitimate decryption of one of the inputs. The subset is selected pseudo-randomly in such a way that no complete paths are revealed.[41] If the cook tries to manipulate votes, there is a high[42] probability it will be caught. This interactive scheme can be collapsed to non-interactive by applying the Fiat-Shamir technique. Randomized Partial Checking was first described by Jakobsson, Juels and Rivest [43].

[41] In order to assure that complete paths are never revealed, the challenger must first pseudo-randomly select which transformations leading to the *last* column of onions should be revealed. After these transformations are revealed, the choice regarding the remaining transformations becomes unequivocal. Readers who want justification for this claim are encouraged to simulate this selection on paper.

[42] In the depicted scheme a single vote could be manipulated without detection with a 50% probability, but the scheme can be trivially altered to provide much higher probabilities.

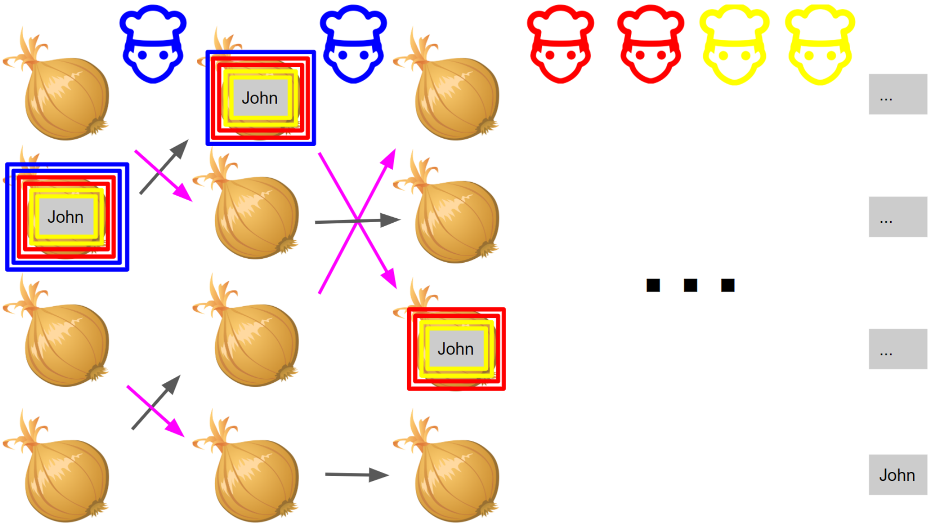

Figure 3: Illustration of randomized partial checking (in the context of a decryption mix network). After the blue cook has performed two rounds of decryption and shuffling, it is challenged to reveal half of these transformations (illustrated as purple arrows). Note that the purple arrows have been selected pseudo-randomly in such a way that no complete paths are revealed. This illustration is original work with visual assets from freeiconspng.com and icons8.com.

5 Case reviews of voting schemes

We wanted to compare a wide range of different kinds of schemes. We review one traditional in-person paper voting scheme (Finland) and one traditional remote paper voting scheme (Switzerland). Other schemes that we review could not be called traditional. We review three remote e-voting schemes which have real-world impact in Estonia, Switzerland and Australia. We also review two academic in-person paper voting schemes (Floating Receipts, Prêt à Voter) and two academic remote e-voting schemes (Helios, Civitas). We usually describe several variants of each scheme, but in the end we choose only one variant for the comparison. Our intent was to select the best variant of each scheme, but opinions may vary.

5.1 In-person paper voting in Finland

We present the Finnish paper voting scheme as a baseline for comparison.[43] The scheme is roughly as follows: The voter walks into a polling station and shows their identification documents to poll workers. A poll worker attempts to confirm the authenticity of the documents, eligibility of the voter to vote and whether the voter has already voted (possibly using printed lists of eligible voters where a voter's name is crossed over after their vote is dropped in the ballot box). The poll worker then hands the voter a ballot and the voter goes to a polling booth to cast their vote on the ballot. Before leaving the booth the voter seals their vote by folding it (similar to sealing a vote inside an envelope). Then the voter walks back to the poll workers and has the option to spoil the ballot or cast it. If the voter spoils the ballot, they have the opportunity to re-vote. If the voter decides to cast the ballot, it is stamped by the poll worker and placed inside the ballot box.

Finland uses (several variants of) the sort-and-stack method [32] for counting. A preliminary count is conducted immediately after the polls close [70]. The ballots are reorganized in stacks where each stack represents votes for a single candidate. Later, these stacks are transported and a final count will be conducted by the local authorities [72]. The final count overrides results from the preliminary count. Discrepancies between these counts are commonplace and are not usually investigated (the people performing the preliminary count are advised not to waste time correcting small errors) [70]. In other words, only the final count matters.

Anti-forgery and chain-of-custody methods are utilized to maintain integrity of the physical ballots. Methods to verify integrity of *the count* are not specified at the federal level [70][72]. Instead, local authorities can choose how they conduct counting of the ballots [75]. In Helsinki, each stack of votes is counted separately by two persons [75]. If the results for a stack differ, both officials count the stack again until they arrive at the same conclusion [75]. The officials rely on their memory to arrive at the correct count for a stack of votes [75], so there is some risk that two officials could accidentally arrive at the same incorrect result. The results are marked on a paper spreadsheet [75], so there is some risk that the results could be accidentally or intentionally marked to the wrong candidate. Later, the paper spreadsheets are manually inputted on a computer [75], opening up additional risks for accidental and intentional manipulation of results.

[43] We describe the 2018 Presidential election variant of the Finnish paper voting scheme. Elements related to voting are described in [71] and elements related to counting votes are described in [70] and [72].

> *It's not the people who vote that count, it's the people who count the votes.*
>
> – Joseph Stalin

The authorities do not produce evidence that citizens could evaluate in order to gain confidence in the reported election outcome (as we noted earlier, compliance audits and risk-limiting audits would produce an evidence trail that we consider sufficient). Regular citizens are not allowed to observe the counting process in neither the preliminary counting stage nor in the final counting stage [75]. However, political parties and the Ministry of Justice can appoint observers to the counting process [75].

Finland has relatively low corruption [99]. Known instances of vote manipulation are extremely rare and the electoral process is highly trusted by citizens. However, a similar scheme can fail spectacularly in a different environment. For example, the paper voting system in India suffered from rampant vote manipulation in the 1970s. Criminals took over entire polling stations, chased legitimate voters away and stuffed ballot boxes [26]. Manipulation committed by election officials are also commonplace in many parts of the world [13][36]. We mention these events to illustrate that although traditional paper voting appears to work in Finland, it is a brittle scheme.

Security properties of in-person paper voting in Finland

We present security properties of voting schemes in a structured manner in order to facilitate comparisons (section 6). Each property has a name in italics and the trust conditions under which it holds. For example, the name of property P3 is *Voter is able to keep their ballot as secret* and we might say that under a particular scheme it holds when "at most one authority is misbehaving". After that we usually give some justification for this claim. Although names of properties are intuitive one-sentence descriptions of properties, they should be treated as references to section 3 where these properties are defined (section 3 contains important clarifications, assumptions, definitions and context for these properties in order to avoid misunderstandings).

P1. *Malware on voting device is unable to violate ballot secrecy.* Always holds, because the voting device is not a computer.

P2. *Malware on voting device is unable to manipulate votes.* Always holds, because the voting device is a not a computer.

P3. *Voter is able to keep their ballot as secret.* Always holds, due to physical properties of the voting booth and the ballot box. Ballot numbering[44] is not used in Finland. We discuss assumptions related to physical security in section 3.3.1.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Holds as long as only few authorities are misbehaving. The voter may attempt to convince a vote-buyer by taking a photograph of a filled ballot, but the voter also has the ability to cheat by spoiling the ballot and requesting another one. Local authorities may collude with a vote-buyer to write down names of voters who spoiled ballots (in order to validate photographs of voters who did not spoil ballots). However, in order for a vote-buyer to successfully buy votes at large scale, the vote-buyer would have to corrupt several local authorities. (Although the vote-buyer could send random people as spectators, they would find it extremely difficult to surreptitiously authenticate which voters are spoiling ballots and which are not. The poll workers, on the other hand, have direct access to this information.)

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Holds as long as only few authorities are misbehaving, due to similar arguments as above.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds. The spouse can accompany the voter to the polling place where they can observe that the voter requests only one ballot from poll workers. This observation, together with a photograph of the filled ballot, can convince the spouse exactly how the voter voted.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to same arguments as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Never holds, because the voting scheme has no mechanism to prevent accidentally spoiling ballots. We refer to figure 1 for an example of an accidentally spoiled ballot.

P9. *Voter can ensure their vote is recorded as cast.* Always holds, because voters are physically placing their votes in the ballot box.

[44] Many countries, including the U.K., uses ballot numbering for audit purposes [76]. These ballot numbers directly link each voter to each ballot, allowing a corrupted authority to break ballot secrecy at will.

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Holds when none of the authorities are misbehaving. Re-voting is not allowed in the scheme. Pre-emptive replacement would be detected by the voter as they walk into the voting place and a poll worker tells them that they have already voted. However, a corrupt local authority could make some individual ballots disappear. No compliance audit exists to verify chain-of-custody for casted ballots.

P11. *The tally is counted correctly from recorded votes.* Holds when none of the authorities are misbehaving. Vote counting is performed by many small groups (”authorities”) and a single corrupt authority will be able to manipulate the tally (for example, by miscounting votes). Although votes are recounted by different officials, the latter count simply overrules the earlier count; discrepancies are usually not investigated. Authorities do not produce convincing evidence for the public (such as compliance audits and risk-limiting audits). The scheme involves no verification methods (such as universal verifiability or individual verifiability). A study has shown that counting methods similar to those used in the Finnish scheme produce an error rate in the order of 1% [32], so one might argue that this property never holds in the Finnish scheme. However, the exact counting methods used in the Finnish scheme are in some ways different compared to the methods reported in the study, so we decided to give the Finnish scheme the benefit of the doubt and assume that honestly behaving authorities are able to arrive at the correct count.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. Methods to prevent ballot stuffing are not sufficient. A local group of poll workers (”1 authority”) could, for example, allow their friends to vote multiple times. No eligibility verification methods exist to reveal localized cheating. No compliance audit exists to verify chain-of-custody of physical ballots and the ballot box (or if some exist, its results are not published to the public).

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. No applications are involved so application-layer DoS vulnerabilities do not exist. However, misbehaving authorities could, for example, close polling places early under a pretense of perceived threats.

5.2 In-person paper voting with Floating Receipts

Floating Receipts was first described by Rivest and Smith [80], who explained it as a standalone scheme (which they refer to as *Twin*) and, alternatively, as a modification to other schemes, such as ThreeBallot[45]. The variant we consider here is essentially Twin with amendments.

We will introduce the idea behind Floating Receipts with a quick thought experiment. Suppose we have a traditional in-person e-voting scheme where each voter inputs their choice on an electronic voting device, and the electronic system opaquely counts the votes. Suppose we modified the scheme so that each voter can take home a paper receipt of their vote. The receipt contains a unique identifier, the choice of the voter, and a digital signature by the authorities. After the election voters can compare their receipts to votes listed on a public bulletin board. In case of discrepancies, they can use the digital signature on their receipt to prove manipulation. The obvious downside is that voters can use the receipts as proof to vote-buyers and coercers.

Now imagine a small tweak: instead of taking home their own receipt, each voter takes home *somebody else's* receipt. The probability that manipulation will be caught does not change (assuming voters still verify the same proportion of receipts), but voters can no longer use the receipts as proof of how they voted (since they show how somebody else voted). Additional constructions are necessary in order to defend against various attacks, but this is the core idea. Next we will motivate and explain the additional constructions.

Suppose there was only a single copy of each receipt. This would mean that an adversary who buys receipts from voters or scours trash bins for abandoned receipts would be able to identify a set of votes which no-one else will be able to verify (since the adversary holds the only copy of the receipt). If this adversary colluded with the bulletin board, it would be able to selectively manipulate votes without detection. As a solution against this threat, Rivest and Smith [80] propose that an unpredictable number of copies should be made of each receipt. This way, even if an adversary finds abandoned receipts in the trash, they will take a risk if they modify the corresponding votes on the public bulletin board, because additional copies of those receipts may exist.

A practical implementation of these ideas involves a semi-transparent bin with receipts and a copying machine. The first few dozen voters simply leave their receipts in the bin without taking anything. Once there is a sufficient amount of receipts in the bin, subsequent voters will take a random receipt from the bin, copy it with the help of poll

[45] We cover ThreeBallot towards the end of section 5.2.

station workers, put the original receipt back in the bin, and put their own receipt into the bin.

Scratch strips

Suppose a voting device prints the identifier on the receipt. If this were the case, a corrupted voting device might perform a *clash attack* where multiple voters who vote for the same candidate are given identical receipts (even though only one corresponding vote is recorded for all those voters). In addition, voters could photograph their receipts in the privacy of the voting booth and later use those photographs to sell their votes. Due to these reasons, the original scheme has identifiers pre-printed on ballots and hidden under scratch strips. The voter can photograph the ballot in the voting booth, but the photograph will not show the identifier because it is hidden under the scratch strip. The voting device is no longer able to perform a clash attack, since identifiers are pre-printed on ballots. The ballot-printing authority is unable to perform a clash attack, since it prints ballots before knowing which ballot will be used to vote for which candidate (if it were to print the same identifier on multiple ballots, it would very likely be caught during verification of receipts).

The authors omit details regarding how, where and when a vote is recorded. The article [80] implies that the vote is recorded by a device in the voting booth. The device confirms that the scratch strip is intact before recording the vote. After recording the vote, the voter walks out of the voting booth and towards the bin. The voter and a poll work together remove the scratch strip from the ballot just before it is dropped into the bin. This interpretation of the scheme has some glaring weaknesses. For example, the voter could record their vote in the booth, but remove the scratch strip and photograph the ballot before leaving the voting booth. In this case the poll worker would notice that the scratch strip has been tampered with, but the voter could plausibly claim it was an accident – and in any case, the vote would have already been recorded.

In order to prevent this vulnerability, we assume that the vote recording device is outside the voting booth. We assume that the voter walks into the booth to mark their ballot with a pen, folds the ballot to protect it from prying eyes, walks out of the voting booth and inserts the ballot into a vote recording device. We assume that the voting device has the ability to unfold the ballot and remove the scratch strip (or, alternatively, that the voter can do this without exposing the ballot to surrounding people and without having the ability to memorize the identifier). Once the vote recording device has accepted the ballot, it signs the ballot with a digital signature, the voter has a possibility to authenticate the signature, and after that the ballot is dropped into the bin.

Another issue arising from scratch strips is the following: if the vote recording device records the vote while the identifier is hidden under the scratch strip, the vote recording device can not possibly know which identifier to post on the public bulletin board later. The ability to do this is crucial to provide verifiability of receipts. This issue could be solved by printing another identifier on the ballot, such as a barcode. The voting device could read the barcode and match it to the hidden identifier by looking up a secret, pre-generated list of pairs. The variant in our comparison includes such a bar code.

Cast-as-intended amendment

The original article [80] does not explain how the voter can perform cast-as-intended verification. Suppose the voter selects ”John” on the voting device. The device then asks ”Are you sure you want to vote for John?”. The voter selects ”yes” and the device says ”Your vote for Sally has been recorded” and prints a receipt which verifies as a vote for Sally.

There are many ways to prevent this. One option is that we have two devices: one device fills the ballot and another device reads it (after the voter has verified the ballot).

Another option may be that the voter fills a paper ballot by hand, which is then scanned by the voting device. The device would warn the voter if the ballot is smudged, undervoted, or otherwise malformed. If the device accepts the ballot and signs it, there is very little the device could do at this point to deceive the voter.[46]

The variant in our comparison includes the latter solution: the voter fills a paper ballot by hand and scans it with the voting device, which refuses to accept ambiguously filled ballots.

Amendment to increase the amount of receipts verified

Another issue with Floating Receipts is that most voters will be unlikely to bother with the verification procedure. We know from experience

[46] Theoretically, the device could display the voter one interpretation of the ballot while actually recording a different interpretation. However, since the voting device already declared the ballot to be unambiguous (by accepting it), the receipt corresponding to this vote would serve as proof that the voting device misbehaved (not immediately, but later when the receipt is verified). The authorities have some plausibility in claiming that the errors were caused by something other than malice, but if the errors are unevenly distributed (to affect the outcome), that plausibility fades away.

that voters generally do not bother verifying their own votes when such verification is possible [8], so we can guess that voters will be even less likely to verify someone else's vote.

We propose the following amendment: voters who do not wish to participate in auditing the election should not be forced to take home any receipts. Instead, voters should be asked if they want to participate or not. For each voter who does not want to participate in the audit, a copy of a random receipt would be made and placed in a "helper organization bin". Each polling place would have several of these bins, depending on how many helper organizations are willing to assist in auditing the election. At the end of the voting day, the helper organizations would take home these receipts and verify them once possible.

With this amendment, significantly more receipts would be verified (instead of being trashed by voters who were forced to take them). We note that this idea is not particularly novel. Rivest and Smith [80] already discuss the role of helper organizations in possibly verifying the receipts which are left in the main bin at the end of the voting day. This amendment can be considered as a natural extension of that idea. The variant in our comparison includes this amendment.

Security properties of in-person voting with Floating Receipts

P1. *Malware on voting device is unable to violate ballot secrecy.* Almost always holds. Although malware on the voting device learns the voter's choice, they never learn the voter's identity (because the device is shared across all voters at a polling place). This relies on certain physical security assumptions discussed in section 3.3.1. We might imagine a scenario where multiple authorities collude in order to break ballot secrecy. For example, the local authorities could write a log with voting timestamps and voter identities and correlate this log to timestamps recorded by the voting device.

P2. *Malware on voting device is unable to manipulate votes.* Always holds. We assume that a significant fraction of receipts will be verified. If malware on the voting device attempted to manipulate more than a few votes in any manner, it would be incredibly likely to be detected.

P3. *Voter is able to keep their ballot as secret.* Always holds. Even if all of the authorities are corrupted, they will not be able to link votes to voters. We discuss assumptions related to physical security in section 3.3.1. Malware on the voting device is considered separately in P1.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Always holds. Although the voter takes home a receipt, it will be a copy of someone else's receipt. The voter might photograph their actions in the booth, but will be unable to prove to a vote-buyer that the ballot will be casted (because the identifier on the ballot is hidden under a scratch strip and the voter might spoil the ballot after photographing it and request another one). If the voter prematurely removes the scratch-strip to photograph the identifier, the voting device will refuse to accept the vote. If the voting device is corrupted and accepts all ballots regardless of scratch strip, the poll workers would notice this. The poll workers could collude with the voting device, a large-scale vote buyer, and voters who wish to sell their votes. However, this scenario seems infeasible, since a single honest voter who refuses to sell their vote would be able to easily prove that the machines have been tampered with.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Always holds, due to same reasoning as above.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds. The voter can photograph their ballot in the booth and the spouse can physically observe that the voter only requests one ballot from the poll workers.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to same reasoning as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Always holds. The voting device refuses to accept ambiguous votes. The voter can ask for a fresh ballot if they accidentally spoiled their ballot. (In addition, the voting device can also confirm the user's choice to them before recording the vote. As we discussed earlier, a corrupted voting device could cheat the voter by displaying a different interpretation of the vote on the screen while sending a manipulated interpretation to be recorded. However, these would be detected in verification and it would be extremely difficult for the voting device authority to claim that such errors were non-malicious when the voting device is supposed to outright refuse any ambiguous ballots.)

P9. *Voter can ensure their vote is recorded as cast.* Always holds (probabilistically), because voters can verify that a vote on a receipt corresponds to a vote on a public bulletin board. We note that with this property and P10, it would be nicer if voters could verify their own votes rather than someone else's. However, we use probabilistic guarantees in many other places as well and see no reason to refuse a probabilistic guarantee for these properties (even though it "feels" wrong). Clash attacks by voting devices are prevented by pre-printing ID numbers on ballots and hiding them under scratch strips. If the voter is unable to locate their take-home receipt on the public bulletin board, the voter can prove manipulation with the digital signature on the receipt. A single missing vote is sufficiently strong evidence of manipulation to invalidate the election, thus satisfying dispute resolution.

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Always holds. Replacing votes is not possible, because no re-voting is allowed. Pre-emptive replacement would be detected by the voter as they walk into the polling place and a poll worker tells them that they have voted already. Deletion of votes is very likely to be caught by the verification mechanism (if authorities delete more than a few votes).

P11. *The tally is counted correctly from recorded votes.* Always holds. Anyone can count the tally from the votes posted on the public bulletin board.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. Misbehaving poll workers would be able to add votes and mark them to voters who did not vote.

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. No known DoS vulnerabilities. Any authority would be able to deny service at will.

Justification why we did not review ThreeBallot/VAV

Readers who are familiar with voting literature may be wondering why we did not review ThreeBallot or VAV. These schemes were described in the same article [80] as Floating Receipts. ThreeBallot (or its generalized variant VAV) typically gets much more attention in literature than Floating Receipts. In fact, we were unable to find any later articles which discuss Floating Receipts at length. This was extremely surprising given the magnificent security properties that Floating Receipts provides.

We acknowledge that ThreeBallot is a fascinating and extemely novel idea. However, it relies entirely on a trusted authority: if the public bulletin board is corrupted, it can collude with voters to manipulate the tally. These collusive attacks were alluded to in the original article [80] and later explained thoroughly by Küsters, Truderung and Vogt [52]. We omit details due to lack of space but we would like to highlight that these attacks can not be detected by outsiders and they fundamentally break the entire scheme.

Rivest and Smith [80] note that *"one can add Floating Receipts as an extra security feature [...] against a wide class of collusive attacks"*. This is true. However, if we compare the simple Floating Receipts scheme to ThreeBallot with Floating Receipts, the former is simpler and has better security properties. In other words, if we are already using Floating Receipts, the incorporation of ThreeBallot is actually a detriment, not an improvement.

It is unclear what advantages – if any – the ThreeBallot scheme with Floating Receipts has over the simple Floating Receipts scheme. We argue that the major contribution in [80] was actually Floating Receipts, not ThreeBallot. We are dumbfounded as to why Floating Receipts has received so little attention in literature compared to ThreeBallot.

5.3 In-person paper voting with Prêt à Voter

We have presented one traditional paper voting scheme (Finland) and one verifiable paper voting scheme which does not heavily rely on cryptography (Floating Receipts). In this section we present Prêt à Voter, a verifiable paper voting scheme which relies heavily on cryptography. Prêt à Voter was published in 2005 by Chaum et al. [17]. A variant of Prêt à Voter named "vVote" [21] has been field tested in real elections in Australia with promising results, but it is not currently used in real elections anywhere.[47]

Ballots in the scheme appear similar to traditional paper ballots: their left side contains candidates, and the voter can mark their preference on the right side. However, the similarities end there. Candidate order on ballots is pseudo-randomly generated for each ballot. Although the candidate order is plainly visible on the left side of the ballot, it is also visible in encrypted form on the right side. A ballot is illustrated in figure 4.

The voter chooses a ballot from a stack of pre-generated ballots and walks into a voting booth. The voter proceeds to mark their choice, tears the ballot in half, and scans the right half with a vote recording device in the booth. If the vote recording device is corrupted, it might attempt to learn the voter's choice. However, the vote recording device never sees the left half of the ballot, and it does not have the keys required to decrypt the ciphertext on the right half.

[47] As of 28.9.2019, to the best of our knowledge.

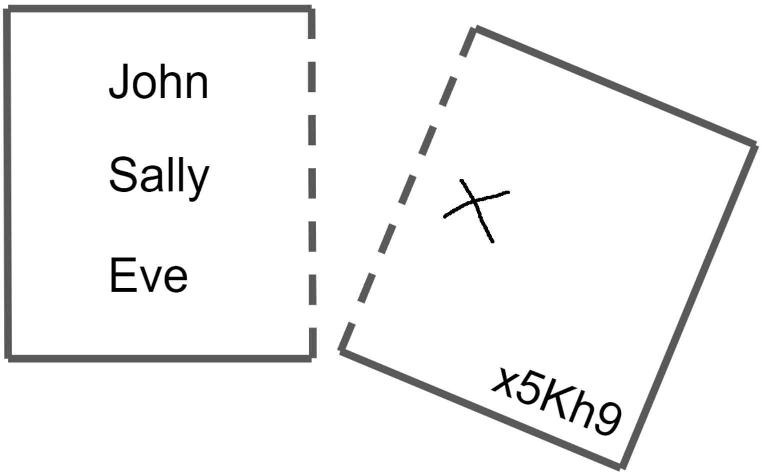

Figure 4: Illustration of a ballot in Prêt à Voter. The voter has marked a vote for Sally with an "x" on the right half, and then separated the halves by tearing. The left half of the ballot contains candidates in a pseudo-random order. Only the right half is recorded for counting. The ciphertext on the right half can be used to reconstruct the order of the candidates. This illustration is a derivation of similar work in [8].

After recording the vote, the voter walks out of the booth, destroys the left half together with election officials, and takes the right half home as a receipt. The voter can later use this receipt to verify that their vote is correctly posted on a public bulletin board: the voter can search the bulletin board with the ciphertext (encrypted ordering), verify that a vote for this ciphertext has been recorded, and that the row number of the vote matches the row number on the receipt. The bulletin board should provide only this information – photographs should be avoided because voters may mark their ballots in unique ways. Note that because of the pseudo-random ordering of candidates, the voter can not use this receipt to prove how they voted.

In order to reconstruct the order of candidates on ballots, the talliers jointly decrypt the ciphertext (utilizing a mix network with randomized partial checking). We find it fascinating that – contrary to usual cryptographic schemes – encryption is not applied to the input of the voter (the chosen row number), but is instead applied to the frame of reference (order of candidates on the ballot).

Improvements to the original Prêt à Voter scheme

Receipt-freeness in the original scheme [17] was completely broken: voters were able to prove their vote by photographing the complete ballot in the privacy of the voting booth. Vote-buyers could then match these photographs to votes on the public bulletin board. We have not seen this vulnerability documented anywhere in literature, but it appears to have been accidentally fixed while fixing a different, unrelated vulnerability.

In [82], Ryan augments the scheme with scratch strips in order to defend against a ”chain voting”[48] vulnerability. In the augmented scheme, scratch strips are used to hide the encrypted candidate ordering. The voter goes to the booth to mark the ballot, then tears the ballot and walks out of the booth. Polling place workers then check that the scratch strip on the right half is intact, remove the scratch strip and scan the right half. This does not violate ballot secrecy, because the left side (human-visible candidate ordering) has already been separated from the right side. Scratch strips fix both receipt-freeness and the chain-voting vulnerability.

The main drawback of Prêt à Voter is that ballots need to be generated and physically handled by trusted authorities. The concern here is related to confidentiality properties, not integrity (the authors describe a ballot well-formedness auditing procedure to probabilistically ensure that the displayed left-side orderings match to the encrypted order on the right side). A corrupt ballot-generating authority or a corrupt polling place worker could collude with adversaries to record the candidate ordering for each ballot, thus facilitating vote-buying and coercion. Ryan and Schneider [83] proposed a new variant of the scheme in 2006, mainly intended to remediate this issue.

Ryan and Schneider’s [83] scheme distributes the construction of ballots over a set of mutually distrusting entities (instead of having a single trusted authority generate the ballots). In their scheme only a part of the ballot is given to the voter at the polling station; the remaining

[48] In a chain voting attack, an adversary first somehow smuggles a single ballot outside of a polling place. This single ballot is later leveraged to coerce an unlimited chain of voters. The first victim is demanded to smuggle this ballot back into the polling place, pick up a fresh ballot, go into the voting booth, fill out the first ballot by voting for a specific candidate, and smuggle out the fresh ballot. Because the adversary has seen the complete ballot, they can later verify from the public bulletin board that the victim voted for the desired candidate. Now the adversary has another ballot and they can repeat the attack on the next victim. An interesting property of this attack is that smuggling out the first ballot is (presumably) very difficult, but smuggling out subsequent ballots is not difficult at all (because poll workers have no idea that people are smuggling ballots into the polling place and swapping ballots in the privacy of the voting booth). [80]

part is generated by mutually distrusting authorities on-demand (when the voter wants to cast their vote).

Unfortunately, on-demand ballot generation requires polling places to have constant electricity, internet connectivity, and availability of the talliers to cooperate in real-time. We do not wish to make such assumptions for in-person elections.[49] Even power generation alone would be prohibitively expensive to guarantee.[50]

We do incorporate one improvement from [83], however. In the original scheme [17] the candidate ordering is reconstructed with a decryption mix network which provides an n-out-of-n quorum. Ryan and Schneider's [83] variant replaces the decryption mix network with a re-encryption mix network, which provides a k-out-of-n quorum. The k-out-of-n quorum is preferrable due to availability issues, so the variant we present in the comparison has this improvement.

Security properties of in-person paper voting with Prêt à Voter

P1. *Malware on voting device is unable to violate ballot secrecy.* Almost always holds. Malware on the vote recording device will see which position on the ballot was marked by the voter, but malware does not know which position corresponds to which candidate. In order to have this information, an adversary who has corrupted the voting device would have to collude with k-out-of-n talliers. [51]

P2. *Malware on voting device is unable to manipulate votes.* Always holds. If the vote recording device modifies the vote in any way, the voter can see this on the public bulletin board and prove the manipulation with the digitally signed take-home receipt.

[49] This may sound odd, given that we make similar assumptions for remote e-voting. However, in the case of remote e-voting, the voting period is typically longer and the harm to voters from outages is smaller.

[50] Ballots need to be printed on a special kind of paper, which requires special kind of printers, which consume massive amounts of power. If we require elections to continue even in case of power outages, we need to invest an infeasible amount in backup power generators. [5]

[51] One might argue that a vote recording device is not a voting device. Voting device in this case would be a pen and a vote recording device would be an unrelated device which should be assessed in other properties. All of this would be technically correct, but we feel that the distinction is rather small, so we chose to view things differently in order to produce a more structured analysis. Note that this choice does not lead to different conclusions, it only leads to a different way of presenting the same conclusions.

P3. *Voter is able to keep their ballot as secret.* Holds as long as at most one authority is misbehaving. A corrupted local authority or a corrupted ballot-generating authority could record candidate orderings on ballots. However, neither a local authority nor a ballot-generating authority has records on which voter casted which ballot. In order to produce such records, a local authority would have to collude with the server which is receiving votes (the local authority could write down timestamps of when specific voters casted votes, and later correlate these timestamps to server logs). (This also explains why votes can not be posted to the public bulletin board in real time.)

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Holds as long as none of the authorities are misbehaving. A corrupted local authority or a corrupted ballot-generating authority could record candidate orderings and collude with an adversary to coerce voters or buy votes. Voters would send their take-home receipts as proof of following the adversary's demands. The adversary would compare these receipts to their known candidate orderings to confirm this.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Holds as long as none of the authorities are misbehaving, due to similar reasoning as above. In addition, voters must be allowed to shift through multiple ballots before choosing which ballots they are going to use – otherwise the scheme becomes vulnerable to randomization attacks. In a randomization attack, a coercer or vote-buyer demands the voter to vote for a specific position on the ballot. For example, the voter may be demanded to vote for whichever candidate is in first position on the ballot that they happen to receive. The receipt the voter takes home can serve as proof that the voter did as requested. When the voter is allowed to shift through multiple ballots, the voter can mitigate this attack (against adversaries who are not physically present) by searching for a ballot which has their preferred candidate in the preferred position.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds. The spouse can accompany the voter to the polling place where they can observe that the voter requests only one ballot from poll workers. This observation, together with a photograph of the filled ballot, can convince the spouse exactly how the voter voted.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to similar reasoning as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Never holds. Although the vote recording device could refuse to scan a ballot which is undervoted or overvoted, it can not verify that the voter has marked their intended candidate as opposed to some other candidate. Remember, with the scratch strip improvement, the vote recording device operates in a public space, so it can not display contents of a secret ballot. If the vote recording device was to be moved back into the private voting booth, then we would have a different problem: voters would be able to photograph complete ballots as proof for vote-buyers. Cast-as-intended verifiability appears fundamentally incompatible with Prêt à Voter.

P9. *Voter can ensure their vote is recorded as cast.* Always holds. The voter can take the right half of the ballot home as receipt and later confirm that it appears on the bulletin board unaltered. In case of any discrepancies the voter can use the digital signature on the receipt to prove manipulation.

P10. *Voter can detect if their vote is displaced (deleted, replaced or preempted).* Always holds. The scheme does not enable displacing a vote by re-voting. Displacing a vote by pre-emptively voting is possible, but the voter can detect this by walking into the polling place and hearing from poll workers that someone has already voted in their name. Deletions from the bulletin board would be noticed by helper organizations who are monitoring the bulletin board for changes.

P11. *The tally is counted correctly from recorded votes.* Always holds. Ballot well-formedness auditing procedures are sufficient to prevent a corrupted ballot-generating authority from malforming ballots without detection. Voters can partake in these audits even if all authorities are corrupted. Tallying is performed with a re-encryption mixnet by a k-out-of-n quorum, who also produce proof of correctness by using randomized partial checking. Even if all talliers are colluding, they are unable to produce proofs for a manipulated tally.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. The scheme does not provide eligibility verifiability. Corrupted polling place workers could add fraudulent votes to voters who did not vote. The authors suggest an auxiliary paper audit trail of cast votes and operational security measures to prevent ballot stuffing. These measures would certainly make ballot stuffing more difficult, but a single corrupted authority could still engage in ballot stuffing.

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. No denial-of-service vulnerabilities are known in this specific variant of Prêt à Voter.

5.4 Remote paper voting in Switzerland

We present the Swiss remote paper voting scheme as a baseline for comparison. Switzerland is an interesting case because all of its citizens can partake in elections and referendums via remote voting (in most other countries remote voting is restricted to absentee voters or other specific small groups). Postal voting is of particular interest, because the majority of voters in Switzerland send their votes by mail [95].

The general idea behind remote paper voting is the "double envelope": Outer envelope has the identity of the voter for authentication purposes. Once it is removed, the inner envelope contains no link to the original voter and protects secrecy of the vote. We were able to find only little information specific to the Swiss remote paper voting scheme.[52] It would appear that the scheme relies heavily on trusted authorities and there are no safeguards against vote-buying and coercion. In particular, re-voting is not allowed.

Security properties of remote paper voting in Switzerland

P1. *Malware on voting device is unable to violate ballot secrecy.* Always holds, because the voting device is a not a computer.

P2. *Malware on voting device is unable to manipulate votes.* Always holds, because the voting device is a not a computer.

P3. *Voter is able to keep their ballot as secret.* Holds when none of the authorities are misbehaving. For example, a disgruntled worker could open both envelopes to violate ballot secrecy.

[52] https://www.ch.ch/en/demokratie/votes/how-to-complete-a-ballot-paper-correctly/ (accessed 1.8.2019)

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Never holds. The voter can, for example, record a video of filling the ballot and dropping it in a mailbox.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Never holds, due to same reason as above.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds, due to same reason as above.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to same reason as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Never holds. No method exists to prevent accidentally spoiling the ballot.

P9. *Voter can ensure their vote is recorded as cast.* Never holds. No method exists to confirm to the voter that their vote is recorded.

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Holds when none of the authorities are misbehaving. If an adversary was able to forge balloting materials, they could pre-emptively replace votes. However, we assume that it would be infeasible for an adversary to forge balloting materials without colluding with the authority. Corrupt authorities (including a single disgruntled employee) could easily destroy envelopes.

P11. *The tally is counted correctly from recorded votes.* Holds when none of the authorities are misbehaving. For example, the authority responsible for opening the inner envelope could easily change its contents.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. No mechanism exists to verify that ballots' credentials correspond to legitimate credentials on a public voter roll.

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. The authority can deny service at will. We do not consider physical mail to be vulnerable to application-level DoS attacks.

5.5 Remote e-voting in Switzerland

Several different remote e-voting systems have been used; we refer to [74] for a historical perspective. However, due to sunsetting of older systems, only one system will be relevant in the near future.[53] We analyze the voting scheme corresponding to this system, officially named *sVote*, often referred to as "the Swiss Post system". The system is developed by commercial vendor Scytl in collaboration with Swiss Post and is currently in use in several Cantons (regions of Switzerland).[54]

Clarifications

This section will be helpful to readers who intend to venture into literature regarding the Swiss Post system. Other readers are encouraged to skip this section.

The Swiss Post system should not be confused with voting systems referred to as "Geneva" or "Neuchâtel" in literature. Although the Canton of Neuchâtel is currently using the Swiss Post system and Geneva might also use it in the future, articles in literature which use these terms are referring to different (sunset) systems.

As noted by Locher et al. [59], details of the sVote scheme are scattered across numerous conflicting documents. We were unable to find a high level description of the scheme in academic articles, on the Swiss Post website, or on Cantons' websites. We made our best effort to reconstruct the scheme from various sources (including other academics' interpretations of the scheme). We note that our reconstruction may contain errors. Furthermore, many versions of the sVote scheme exist. We attempt to reconstruct the latest version of the strongest variant of the scheme (often referred to as the "100%" version, as opposed to the "50%" and "30%" versions).

Swiss regulators, Scytl and Swiss Post attempt to re-define individual verifiability. Typically the term refers to voter's ability to verify that their vote has been appropriately counted. However, the Swiss meaning for this term sometimes[55] assumes that the server side voting software is trusted. If the server is trusted, individual verifiability only protects against fault on the client side, so the voter can not verify that their vote has been appropriately counted (this assumption does not exist in the "100%" variant). Similarly, the Swiss attempt to re-define universal

[53] https://www.swissinfo.ch/eng/digital-voting_geneva-shelves-e-voting-platform-on-cost-grounds/44577490 (accessed on 11.8.2019)

[54] https://www.post.ch/en/business-solutions/e-voting/success-through-cooperation#partnercantons (accessed on 11.8.2019)

[55] In the case of the "50%" variant, the server is trusted [87]. In the case of the "100%" scheme, the server is not trusted [86].

verifiability [74]. Although the term usually refers to a verification that is available to anyone, the Swiss consider it to be a verification that is available to privileged insiders from Cantons.

Swiss regulators also coined the term *complete verifiability* in reference to the union of individual verifiability and universal verifiability [74]. This terminology is unfortunate on many levels. First of all, as we noted earlier, the Swiss definitions for individual and universal verifiability are drastically different from definitions established in literature.

Secondly, even if stricter definitions for individual and universal verifiability were used, they are insufficient to provide complete verifiability in any meaningful sense of the word. For example, outcome of the election can be manipulated without detection by ballot stuffing when a voting scheme provides no eligibility verifiability. Since the verification processes in use are insufficient to verify that the outcome has not been manipulated, clearly it is inappropriate to refer to these verification processes as providing ”complete verifiability”.

Exacerbating these issues is the fact that Scytl erroneously claims the concept of ”complete verifiability” to be ”equivalent” to ”end-to-end verifiability” [74].

Scheme description

First, a trusted authority sends each voter a voter card by mail. The voter card contains 4 items: pseudonymous credentials for logging in, ”return codes”, ”ballot casting key” and ”vote cast code”. The voter logs on the voting website with the pseudonymous credentials and inputs their choices in plain text. A client side web application encrypts the choices before sending them to the server. After the voter sends their choices, the server responds with a return code for each choice on the ballot. The voter compares the return codes on the screen to return codes on the voter card to confirm that their choices are cast as intended. The voter then inputs their ballot casting key and the server responds with a vote cast code. If the vote cast code matches to the one on the voter card, the voter can be confident that their ballot was recorded as cast. [95][97][86]

At the end of the election, votes are decrypted with a verifiable mix network which produces zero-knowledge proofs of correctness. Each server in the mix network is running a different operating system and different software developed by independent teams. However, all of the servers are developed and operated by the same vendor, Scytl. Furthermore, the zero-knowledge proofs of correctness are not released to the public – only to Canton authorities. [97]

Discovered vulnerabilities

In 2019 the Swiss Post released portions of their source code as part of a "public intrusion test".[56] Shortly aftwards, researchers discovered that the purported universal verifiability and cast-as-intended verifiability were completely broken [55][56][54]. These vulnerabilities would allow – among many other things – a corrupted authority to manipulate the tally and forge a proof which seemingly proves that the tally is not manipulated.

Swiss Post's response to the disclosures has been abysmal. In the same press release[57] where they note that they are temporarily suspending their e-vote platform to apply some fixes, they are celebrating how the public intrustion test yielded only low-severity vulnerabilities. This is technically correct, because the researchers who reported the critical vulnerabilities were not taking part in the public intrusion test. However, a casual reader might read the press release and get the impression that no critical vulnerabilities were found and that the platform has been suspended to implement minor fixes. This impression is reinforced by the titles in the press release, which include "Ballot box not hacked" and "No manipulated votes".

In another press release[58] (where they admit the severity of the vulnerability before later backpedaling[59] that it is not actually severe at all), Swiss Post reveal that they were aware of at least one of the vulnerabilities in 2017 and asked Scytl to fix it. We wonder how is it possible that Swiss Post and Scytl have not fixed the vulnerability in this time? Furthermore, why did Swiss Post, Scytl, and the Swiss government claim during this time that the system was formally proven secure, when clearly[60] it was not?

According to trustworthy outside researchers [59], Swiss Post and Scytl have made software fixes to remediate the vulnerability and a confidential internal document has been released documenting the fixes. Although we are unable to verify these claims due to confidentiality of the document and lack of public disclosures by Swiss Post and Scytl, we will give Swiss Post the benefit of the doubt in our comparison and

[56] https://www.onlinevote-pit.ch/ (accessed on 12.8.2019)

[57] https://www.post.ch/en/about-us/media/press-releases/2019/swiss-post-temporarily-suspends-its-e-voting-system (accessed on 12.8.2019)

[58] https://www.post.ch/en/about-us/media/press-releases/2019/error-in-the-source-code-discovered-and-rectified (accessed on 12.8.2019)

[59] https://www.evoting-blog.ch/en/pages/2019/new-finding-in-the-source-code (accessed on 12.8.2019)

[60] The proofs published by Scytl may or may not pass mathematical scrutiny. Regardless, they do not prove the claimed security properties of the system (as said properties turned out to be fundamentally broken).

assume that these (implementation) vulnerabilities are not part of the scheme (this is an assumption on our part, because as we noted earlier, the scheme is not properly documented anywhere).

Security properties of remote e-voting in Switzerland

P1. *Malware on voting device is unable to violate ballot secrecy.* Never holds. Voter inputs their choices in plain text on a device which is susceptible to malware.

P2. *Malware on voting device is unable to manipulate votes.* Holds when none of the authorities are misbehaving. If malware attempts to modify the choices on the voter's ballot, the voter will notice this manipulation when they compare the return codes sent by the server to the return codes that were mailed to them beforehand. As long as the authority generating return codes is not colluding with the adversary, the malware will be unable to fake return codes.

Dispute resolution exists: when the voter notices that the return codes do not match, the voter refuses to input their ballot casting key, effectively abandoning the current ballot before depositing it in the virtual ballot box. After this the voter can re-vote by post or in-person. (The voter can not re-vote on another machine. If this was possible, then the malware could simply fetch the correct return codes as a trick to get the voter to input their ballot casting key – then, instead of using the ballot casting key to verify the real vote, the malware could craft a different vote with other choices and use the ballot casting key for it instead.)

An additional threat exists and is accounted for in the design of the scheme: the malware might record and send the vote correctly and then refuse to send the ballot casting key, while pretending to the voter that the ballot casting key is sent. To protect against this threat (and potential bugs and network issues with the same effect), the server responds to the ballot casting key with a vote cast code. The voter again compares the code sent by the server to a corresponding code sent to them by mail beforehand.

P3. *Voter is able to keep their ballot as secret.* Holds when none of the authorities are misbehaving. The software is developed by an exclusive vendor and the scheme is not software independent. Therefore, the vendor could surreptitiously modify the components of the system to collude in order to break ballot secrecy.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Never holds. The voter can convince a vote-buyer by, for example, recording a video of the voting process. The scheme has no mechanism to reduce vote-buying and coercion.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Never holds, due to same reason as above.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds, due to same reason as above.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to same reason as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Always holds. If the voter has accidentally spoiled their ballot, they can detect this by comparing return codes. The voter has dispute resolution in the form of a re-vote by mail or in-person.

P9. *Voter can ensure their vote is recorded as cast.* Holds when none of the authorities are misbehaving. The voter can verify that the server received the vote, but the voter has to trust that the server recorded the vote. Although the scheme has a bulletin board where votes are posted, it appears to be private, not public.

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Holds when none of the authorities are misbehaving, because deleting votes without detection appears to be possible. As far as we can tell from the available documentation, there are no mechanisms that the voter could use to verify that a corrupted authority, such as the exclusive vendor, does not delete their votes. Re-voting is not allowed. Pre-emptive replacement would be detected by the voter in the web interface.

P11. *The tally is counted correctly from recorded votes.* Holds when none of the authorities are misbehaving. Although votes are tallied with a verifiable mix network, the same authority operates all the servers in the mix network. Proof of correctness for the tally is not universally verifiable; only a privileged auditor can verify the proof. Presumably, the exclusive software vendor provides software for both the servers in the mix network and the auditor, so this corrupt authority could manipulate the tally without detection.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. We were unable to find any mechanism intended to prevent ballot stuffing. A corrupted vendor or a corrupted authority that is generating voter cards could generate credentials for non existent voters without detection.

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. The authorities could deny service at will. No application level DoS vulnerabilities are known.

5.6 Remote e-voting in Australia

Several remote e-voting systems have been used in Australia [22][25][11]. In this section we analyze only one of them: the *iVote* scheme used in municipal elections in the state of New South Wales (NSW). We made this choice because of the relatively high impact of the NSW iVote system.[61] This system was developed by commercial vendor Scytl, which also developed the Swiss (Swiss Post) system analyzed in section 5.5. The scheme corresponding to the NSW iVote system is described in detail in [11]. We will provide a short description and some observations.

In the NSW scheme, a voter registers to vote online (by connecting to a trusted server in unsupervised conditions on an untrusted device). The voter receives an iVote Number and PIN. Later, the voter can use these secrets to authenticate to vote on a web client. After voting the voter receives a receipt number for verification purposes. If the voter was coerced or made a mistake, they can re-vote.

The scheme provides two separate verifications. First, a telephone verification where the voter calls to the verification service, inputs the iVote Number, PIN and receipt number, and the verification service decrypts contents of the vote audibly. This verification is available only until the end of the election. After that the second verification becomes available. It involves the voter matching their receipt number on a list of receipt numbers on a public web site.

With regards to re-votes, a voter can use the receipt number to verify after the election that their vote was not displaced, although this verification relies on trusted authorities, just as the phone verification does. And naturally, a vote-buyer or coercer can use these same mechanisms to verify that the voter followed their demands.

[61] When the system was first taken into use in 2015, it was used by over 280 000 voters [25]. Although this represents only 5% of votes cast in the election, it broke records as the highest absolute number of votes cast in an election via remote e-voting [25]. To the best of our knowledge, this system still represents the largest remote e-voting deployment to date, in terms of absolute numbers of votes casted.

Although the creators of the system emphasize [11] that mutually distrusting authorities run different components of the voting system, the scheme actually has many weak points which allow a single corrupted authority to break important security properties. As an example, the same vendor provides software for all authorities. Because the scheme is not software independent, a compromised vendor could manipulate the tally without detection.

As another example, consider the telephone verification service. It is able to connect a plaintext vote with a voter's pseudonymous voting credentials. The connection between these credentials and the voter's identity is obfuscated by a component operated by a different authority, seemingly protecting a corrupted verification service from violating ballot secrecy. However, we can expect verifying voters to use their personal phone numbers to call into the verification service, allowing the service to connect personal phone numbers to plaintext votes.

Halderman and Teague [35] note that the verification mechanisms do not offer dispute resolution. If a voter complains about fraud, the authorities will be unable to distinguish between a voter who has been actually defrauded and a voter who simply lies or made a mistake. Although voters can in some cases rectify the issue by re-voting, only a small proportion of voters can be expected to bother in practice[62]. This means that an attacker, such as a corrupted authority running the core voting system, could selectively drop a large number of votes for an undesired candidate. Voters who detect the error and complain would be content after re-voting successfully, but a large number of manipulations would go undetected. If the voters were able to prove that manipulation has occurred, courts would be able to interfere and halt the fraudulent election. But no such mechanism exists.

Security properties of remote e-voting in Australia

P1. *Malware on voting device is unable to violate ballot secrecy.* Never holds, because the voter inputs their choice as plaintext on a device which is susceptible to malware.

P2. *Malware on voting device is unable to manipulate votes.* Never holds. The scheme has verification and re-voting, which allows an individual voter to correct their own vote by re-voting from another device, but the scheme is missing dispute resolution which would prevent a large-scale malware campaign from modifying a large number of votes from voters who do not verify. See the definition of this property for more details.

[62] Only 1.7% of voters who used the iVote in 2015 used the verification service [8].

P3. *Voter is able to keep their ballot as secret.* Holds when none of the authorities are misbehaving. Ballot secrecy could easily be broken by the software vendor acting on its own, by the verification service acting on its own, by the registration service acting on its own, or by the core voting service acting on its own.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Never holds. The two verification processes together can prove to a large-scale vote-buyer that the voter did as instructed. Both of these verifications can be automated.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Never holds, due to same reasons as above.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds, due to same reasons as above.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds, due to same reasons as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Always holds, due to the verification service.

P9. *Voter can ensure their vote is recorded as cast.* Holds when none of the authorities are misbehaving. If the core voting service is misbehaving, it may drop the real vote while sending the copy to the verification service. However, this would be caught during the audit phase when it would be noticed that the set of copies on the verification service does not match the set of copies that was tallied. This is one area where the architectural design of mutually distrusted components actually brings some value. However, the scheme is not software independent and the same vendor provides software for all the components. Therefore, a compromised vendor on its own could cause all components to collude in a manner where the voter receives confirmation of a vote which is not actually recorded.

P10. *Voter can detect if their vote is displaced (deleted, replaced or preempted).* Holds when none of the authorities are misbehaving. By using both of the verification services, the voter can be convinced that their vote was not displaced, as long as they trust all of the authorities. A compromised vendor could deceive the voter (due to similar reasoning as above).

P11. *The tally is counted correctly from recorded votes.* Holds when none of the authorities are misbehaving. For example, the exclusive vendor could undetectably manipulate the tally (due to similar reasoning as above).

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. For example, the exclusive vendor could undetectably add votes corresponding to non existent voters (due to similar reasoning as above).

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. No known DoS vulnerabilities, but any of the authorities could deny service at will.

5.7 Remote e-voting in Estonia

Estonia is a unique case: all of its citizens have the right to vote in significant governmental elections via remote e-voting. Remote e-voting was first used in real elections in 2005. After that the scheme has been improved several times. A mobile application was introduced in 2013 to provide some aspects of individual verifiability which are mainly intended to defend against compromised client machines [38]. Another interesting improvement was in 2017 with properties similar to universal verifiability [37]. Our analysis is limited to the version currently in use, which according to Estonian authorities[63] is the 2017 version ”IVXV” [94].

Estonia leverages their examplary Public Key Infrastructure in their voting system. Voters use their existing smart cards to authenticate and sign votes. The same smart card can be used for a variety of purposes; one can even withdraw bank loans with it. This serves as a massive deterrent against relinquishing these smart cards to vote-buyers or coercers.

Estonia's approach to ballot secrecy mimics the ”double envelope” used in remote paper voting. Voters send encrypted votes (”inner envelope”) along with a digital signature (”outer envelope”) to authenticate the voters. The server-side components which handle these votes can use the digital signatures to link votes to individual voters, but they do not have the key needed to decrypt votes. The components which are able to decrypt votes are behind an airgap, and all the digital signatures are removed before the votes are delivered for decryption. The decryption key is not distributed across mutually distrusting parties. [94]

[63] https://www.valimised.ee/en/internet-voting/documents-about-internet-voting (accessed 31.7.2019)

This approach has no physical limitations to prevent authorities from violating ballot secrecy (compared to physical ballot boxes, which prevent linking votes to voters by physical limitations of the ballot box). Likewise, this approach has no mathematical properties to prevent authorities from violating ballot secrecy (compared to some cryptographic methods, which provide mathematical guarantees under certain assumptions). While Estonia's approach into ballot secrecy may sound convincing to laypersons, we would summarize it as ”trusted authority promises not to look”. As we noted in section 3, when the same authority operates two different services, we consider those services to represent the same authority.

A smartphone application is offered as a defense against malware on the voting device. The voter can scan a QR code up to 30 minutes after voting to verify that their vote was cast as intended and recorded as cast. Clash attacks are prevented by linking the vote confirmation to the voter's identification. [37]

As a defense against coercion, re-voting multiple times is allowed. Only the last vote is counted, and a paper vote on election day overrides all electronic votes. Potential coercers and vote-buyers may obtain proof that a particular vote was sent, but they can not obtain proof that the vote was not later displaced by a re-vote. The downside of this defense is that voters can not obtain proof that their last vote will be counted appropriately. For example, malware on the voting device can be utilized to later re-vote on behalf of the voter [91].

The original scheme was modified in 2017 with aspirations to provide universal verifiability [37]. Votes are now encrypted with a (non-distributed) El Gamal cryptosystem in order to enable tallying with a verifiable[64] re-encryption mix network. However, the verification is available only to auditors designated by the election organizer. Furthermore, while the verification seemingly provides assurances that each decrypted vote corresponds to a recorded vote, there are no assurances that recorded votes were not replaced, removed, or added surreptitiously.[65]

Springall et al. [91] note that procedures in Estonia's voting scheme appear to be designed for ”happy path” only; any deviations (which may indicate potential vote tampering) are dealt with on an ad hoc basis, sometimes by a single election employee. Springall et al.'s analysis was conducted in 2014, but we can confirm that the official documenta-

[64] According to the official documentation [94], this verifiability is considered optional and may not be provided in every election. No justification for this is provided. For the purposes of our comparison, we decided to assume that this verification is in use.

[65] Some assurances are described in [37], but they rely on a trusted authority (which operates two server components which are assumed to not collude with each other).

tion [94] as of 2019 does not contain procedures outside the happy path. We wanted to highlight this as an interesting observation (the properties we selected for the comparison are not affected by this observation).

Estonian officials erroneously claim in [94] that their system is *”end-to-end verifiable: the input and output of all processes can be verified mathematically”*. Many of these processes are not verifiable and rely entirely on trusted authorities, so the ”input and output of all processes” can not be verified mathematically. For details we refer to our analysis of security properties below.

In summary, Estonia’s voting scheme relies heavily on trusted authorities, but has some useful verifiability and anti-coercion properties.

Security properties of remote e-voting in Estonia

P1. *Malware on voting device is unable to violate ballot secrecy.* Never holds, because the voter inputs their choice as plaintext on a device which is susceptible to malware.

P2. *Malware on voting device is unable to manipulate votes.* Never holds. Although the voter can use the smartphone application to verify their vote after voting, malware can displace the real vote by re-voting after the 30-minute verification window has expired.

P3. *Voter is able to keep their ballot as secret.* Holds when none of the authorities are misbehaving. The same authority runs all components of the voting scheme, so they can decrypt votes at will.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Holds when none of the authorities are misbehaving. If no authorities are colluding with the vote-buyer, the voter can send proof that they voted a specific way, but the voter is unable to prove if that vote was counted or displaced by a re-vote. The same authority runs all components of the voting scheme, so they can decrypt votes at will.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Holds when none of the authorities are misbehaving, due to same reasons as above.

P6. *Voter is unable to prove to their spouse how they voted.* Always holds. The spouse is unable to collude with the authorities, and the voter is unable to prove which vote is their last vote.

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Always holds due to same reasons as above.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Always holds. The smartphone application can verify this.

P9. *Voter can ensure their vote is recorded as cast.* Always holds. The smartphone application can verify that the vote is recorded as cast and the voter has dispute resolution in the form of a re-vote.

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Never holds. No mechanism exists to detect if a vote is displaced by re-voting. This appears to be a conscious design choice in order to avoid coercion. In addition, a corrupted authority could delete some votes without detection.

P11. *The tally is counted correctly from recorded votes.* Holds when none of the authorities are misbehaving. The same authority runs all components of the voting scheme, so they can manipulate the count at will.

P12. *No ballot stuffing.* Holds when none of the authorities are misbehaving. The same authority runs all components of the voting scheme, so they can insert new votes at will. No mechanism exists to verify that ballots' credentials correspond to legitimate credentials on a public voter roll.

P13. *Denial-of-service resistance.* Holds when none of the authorities are misbehaving. The scheme has proven to be resilient against outsiders' DoS attacks in real-world conditions. However, the same authority runs all components of the voting scheme, so they can deny service at will.

5.8 Remote e-voting with Helios

Out of all the cryptographic remote e-voting schemes proposed in academic literature, Helios is the first with a practical open-source implementation. The design philosophy behind Helios can be described as simplicity over complexity and integrity over privacy. Although a verifiable mixnet is used for integrity purposes, there is only a single tallier, the centralized Helios server (so it is simply trusted to maintain the privacy of the votes). Furthermore, Helios does not attempt to solve the coercion problem. As such, it is intended for low-coercive environments, such as student unions, sports clubs, or international organizations [2].

Helios was published in 2008 and as of 2019 it remains the only system of its kind[66] with real-world use. For example, it is currently used for internal elections by the International Association for Cryptologic Research.[67] There are multiple variants of Helios – our analysis is constrained to the scheme described in the original article [2] by Adida.[68]

Next we will provide a short description of the voting process. A voter begins the process by navigating to a webpage which serves the client-side application (in-browser). The voter fills out the ballot and the client side application encrypts it with randomized encryption. The application commits to this encryption by displaying the user a hash of the ciphertext. After the ballot has been encrypted, the voter has two choices: they can either audit the ballot or seal it.

If the voter chooses to audit the ballot, the client side application displays the ciphertext and the randomness used to encrypt the vote. The voter can use a secondary device with third-party software to verify that the randomness can be used to produce the given ciphertext and that the ciphertext hashes to the given hash. This audit can help the voter to gain confidence in the system when it is operating honestly. However, if the audit fails, the voter has no recourse. A client-side software running on their computer did something malicious or erroneous. It would be very difficult for the voter to prove manipulation.

If the voter chooses to cast the ballot[69], the client side application makes a network call to authenticate the voter. If the authentication is successful, the encrypted vote is posted on the public bulletin board. The voter can use a secondary device to confirm that their vote was recorded (by comparing the hash displayed on the device that they voted with). If the vote was not delivered, the voter can try again, possibly on a third device. If the vote was altered or the central server refuses to accept the vote, the voter has no recourse to prove any wrongdoing.

[66] We are not aware of any other academic remote e-voting schemes which have both a practical open-source implementation and also cryptographic verifiability properties. Although such implementations may exist, we likely would have discovered them during our research if they had any real-world use. For example, Civitas has an implementation, but it is not practical, so it is not used anywhere. Many commercial vendors also push their implementations, but these are typically closed source and not verifiable (see appendix C for examples).

[67] https://www.iacr.org/elections/eVoting/ (accessed 13.9.2019)

[68] We would have preferred to analyse the latest version of Helios, but it does not appear to be documented anywhere.

[69] The voter is not able to seal the audited ballot – if they audit a ballot, they must encrypt their vote again with a different randomness. In other words, this audit process is probabilistic. The article does not explain the rationale for this (after all, technologically savvy voters could trivially alter the client side application to allow them to audit and seal the same encrypted ballot).

Security properties of remote e-voting with Helios

P1. *Malware on voting device is unable to violate ballot secrecy.* Never holds, because the voter inputs their choice as plaintext on a device which is susceptible to malware.

P2. *Malware on voting device is unable to manipulate votes.* Never holds. Malware can, for example, steal the voter's login credentials to vote on their behalf. In addition, Helios is vulnerable to clash attacks where the same encrypted vote is presented as evidence to multiple voters (who vote the same candidate) [53].

P3. *Voter is able to keep their ballot as secret.* Holds when none of the authorities are misbehaving. The centralized Helios server is able to decrypt votes at will and it is able to link votes to individual voters.

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Never holds. The voter can trivially prove their vote to a large scale vote-buyer.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Never holds. Helios publishes a list of voters and their encrypted votes, so anyone can always know who abstained from voting.

P6. *Voter is unable to prove to their spouse how they voted.* Never holds. The spouse can physically observe the voter casting their vote. Re-voting is not allowed and the voter is not offered other ways to fool a physical observer (see Civitas for an example).

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds. Helios publishes a list of voters and their encrypted votes, so anyone can always know who abstained from voting.

P8. *Voter can ensure their ballot is not accidentally spoiled.* Always holds. The client side software prevents casting malformed votes by accident.

P9. *Voter can ensure their vote is recorded as cast.* Never holds, because dispute resolution is not satisfied. Helios publishes a list of votes' hashes on a public bulletin board. The voter can compare the hash reported by their voting device to hashes on the bulletin board. (Server-side clash attacks are not possible, because hashes are generated by the client. Client-side clash attacks are considered in P2, not here.) Dispute resolution is not satisfied, because if the voter were to notice that their vote is missing from the public bulletin board, they have no recourse: re-voting is not allowed and they can not prove that the server misbehaved.

P10. *Voter can detect if their vote is displaced (deleted, replaced or preempted).* Always holds. The voter can compare the vote generated by their voting device to votes published on the bulletin board.

P11. *The tally is counted correctly from recorded votes.* Always holds. Anyone can validate the mixnet shuffle proof and decryption proof and recount decrypted votes.

P12. *No ballot stuffing.* Always holds. Helios publishes a list of voters and their encrypted votes. Anyone can trivially verify that counted votes correspond to the list of voters. However, there is no easy way to verify that votes linked to specific voters were actually cast by said voters. The scheme contains an audit where auditors sample random voters and contact them personally to request them to verify their votes. The scheme description implies that the election results could be invalidated if many voters report in the audit that someone has added fraudulent votes to their name. Although we have doubts that real election organizers will undertake such audits, we will treat the scheme as described and assume that these audits are undertaken and that they have the ability to invalidate election results.

P13. *Denial-of-service resistance.* Never holds, due to a vulnerability in the audit procedure (see P12). Disgruntled voters could invalidate the election by lying to auditors that someone has voted in their name. The scheme involves no ability to prove or disprove such claims. To our knowledge we are the first to report this vulnerability. (Note that if this audit procedure is weakened to the point where disgruntled voters can't invalidate an election, then a different property breaks: the scheme becomes vulnerable to ballot stuffing where votes are fraudulently added to voters who did not vote.) In addition, the centralized Helios server could trivially prevent voting and tallying.

5.9 Remote e-voting with Civitas

Civitas is a remote e-voting scheme developed in academia around 2008. What makes Civitas interesting is its coercion-resistance properties. A multitude of Civitas variants have been proposed in literature. We begin with a precursor of Civitas. After that we succinctly describe Civitas' key differences to its precursor. After that we discuss various weaknesses and proposed improvements. In the end we define a single variant for the comparison.

Introduction to JCJ, a precursor of Civitas

The notion of coercion-resistance was formalized by Juels et al. in [45] where they proposed JCJ as a coercion-resistant voting scheme. Previous voting schemes had only provided receipt-freeness, so they were vulnerable to threats such as voters selling their voting credentials or a spouse of a voter standing over their shoulder. JCJ tackles these issues by introducing the concept of forged credentials: voters can forge credentials which will be indistinguishable from valid credentials. When a voter is coerced, they can vote with forged credentials or even relinquish these forged credentials to the adversary. Votes casted with forged credentials will not be counted. To clarify, we have 3 categories of credentials: real credentials (votes will be accepted and counted), forged credentials (votes will be accepted but not counted), and invalid credentials (votes will not be accepted).

JCJ is based on mix network tallying. Voters send encrypted votes to a public bulletin board. This set may contain malformed votes, duplicates (multiple votes with the same credentials), and votes crafted with forged credentials. After voting is complete, the tallying authorities transform this set of votes into a smaller set of legitimate votes, but no-one will be able to tell which vote in the input set corresponds to which vote in the output set. This process is described with the following steps:

1. Eliminate malformed votes. As the voters sent their votes, they also sent proofs which can be used to verify that the encrypted votes are correctly formed. For example, a vote is supposed to contain a value of 0 or 1 for each candidate; if a vote contains value 1000, it will be eliminated. Votes with invalid credentials are considered to be malformed as well (not to be confused with forged credentials, which are accepted at this step).

2. Eliminate duplicates. The talliers perform pairwise plaintext equivalence tests on the credentials of all submitted votes, such that only the first vote for each credential remains. (The credentials are encrypted with randomized encryption). The tallying authority provides proof that this elimination is done correctly. Anyone can find out which votes were eliminated in this step.

3. Anonymization. A re-encryption mix network is used to anonymize votes. The list of legitimate credentials is also anonymized. Zero-knowledge proofs are published to prove that no votes or credentials were altered.

4. Eliminate forgeries. The talliers perform pairwise plaintext equivalence tests between all credentials of votes and the list of legitimate credentials. Votes sent with forged credentials will be eliminated, as they do not match to any of the legitimate credentials. The talliers publish a proof of correctness. Even though the proof identifies which votes had forged credentials, no-one is able to link these anonymized votes back to the input set of votes, because the votes were anonymized in the previous step.[70]

5. Decrypt. Only valid votes remain because invalids, duplicates and forgeries have been pruned. Because the votes have been anonymized, they can be decrypted without violating ballot secrecy. All the talliers co-operate to decrypt votes so that the tally can be computed. The decryption is publically verifiable. Everyone can verify the tally from the decrypted votes.

These steps are illustrated in figure 5.

[70] The amount of votes sent with forged credentials becomes known. This represents a theoretical security risk. For example, if the amount was zero, then anybody who watched someone vote would be able to know that they did not use forged credentials. This risk can be mitigated by asking some voters to send in a random amount of votes with forged credentials.

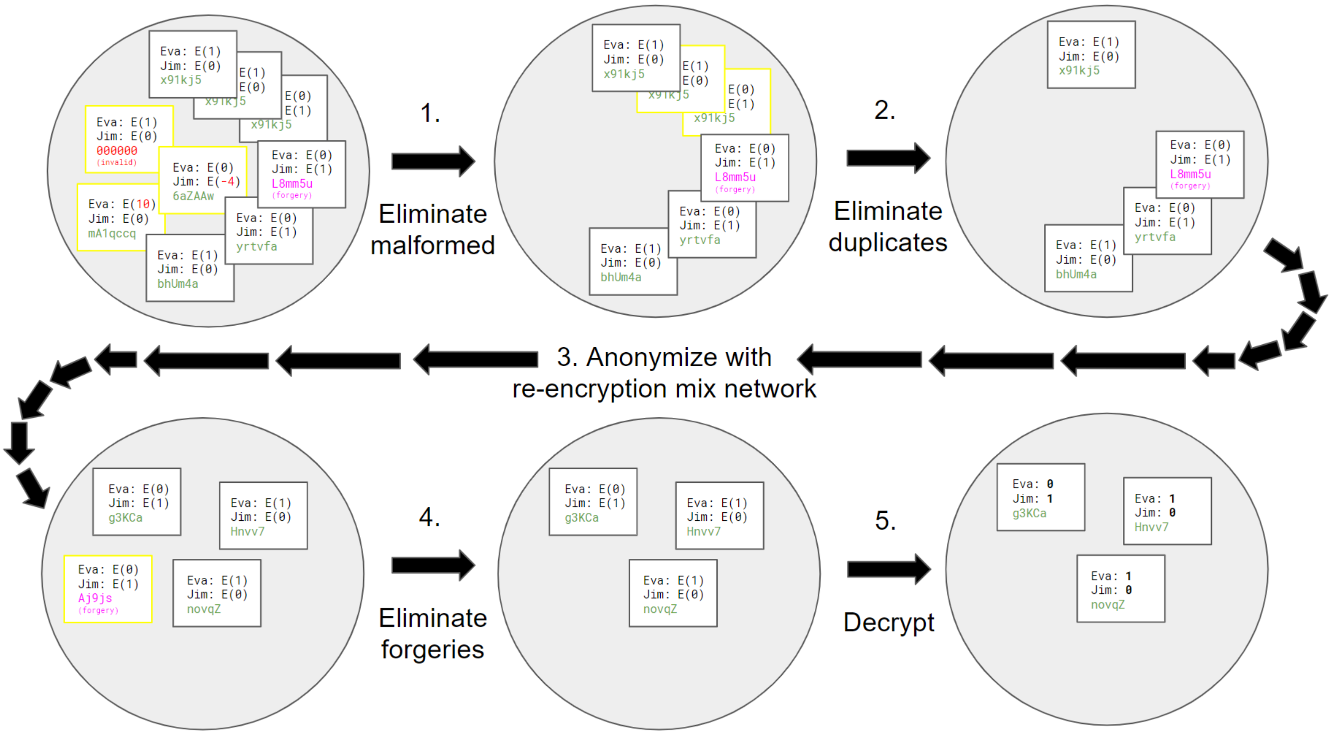

Figure 5: Illustration of the tallying steps in Civitas. Yellow highlight indicates items which are about to be eliminated. Green indicates valid voter credentials. Voters' credentials are always encrypted (re-encrypted in step 3). The actual votes remain encrypted until step 5. E(1) and E(0) are shorthand for randomized encryptions to "1" and "0". This illustration is original work.

JCJ credential verification issues

In JCJ the registrar sends the voter credentials which consist of a corresponding public and private key (similar to typical asymmetric cryptography). The authors treat verification of credentials as an optional element in the voting scheme: "*We assume the trustworthiness of [the registrar] [...] Still, if desired, [the registrar] can furnish [the voter] with a proof that [the public key of their voting credentials corresponds to the private key]. Where erasure of secrets by voters is not automatic, a designated verifier proof is needed for coercion-resistance.*"

As a reminder, the key idea in Civitas is that we do not have a typical public/private key-pair – we have a special kind of key-pair where forgeries of the private key can be made. If the voter can defraud a vote buyer with these forgeries, there is a risk that a corrupt registrar can also defraud the voter with a forgery. Next we will explain why the verification of credentials by the voter has to be mandatory. After that we will explain how the JCJ scheme is insufficient given the mandatory verification of credentials.

Without verification of credentials, a corrupt registrar could pass forged credentials to real voters and their corresponding real credentials to a colluding adversary. Due to the coercion-resistant nature of the scheme, none of the real voters could notice that their votes aren't counted. Thus a corrupt registrar would be able to entirely fabricate an election result without anyone noticing. Due to this threat we consider the verification of credentials to be mandatory.

Suppose verification of credentials is mandatory. The registrar uses designated verifier proofs to prove to the voter – and only the voter – that their voting credentials are valid. The voter is able to generate forged credentials which will be indistinguishable to a vote-buyer (for more information on designated verifier proofs, see section 4). In order for the designated verifier proof to work as designed, the verifier has to know something that the prover doesn't know: a private key which can be used to forge a proof. The problem is, in JCJ the voters only have one private key, and it is generated by the prover. Since the prover also knows the private key, they may as well send a fake key along with a forged proof, since they can forge proofs with the real key. Now the voter has the same issue with the proof as a potential vote-buyer: they are unable to distinguish between a real key and a fake key.

In order to fix designated verifier proofs we need the voters to have a second set of keys, such that the public key is known to the registrar, but the private key isn't. The obvious (although not the only) way to achieve this is with public key infrastructure and proliferation of smart cards. This would be extremely expensive to set up, and as of 2019 only one country (Estonia) has made this investment.

Suppose we have public key infrastructure to support designated verifier proofs. The authors suggest that in practice real credentials would be physically mailed and forged credentials could be produced by a computer program running on a voter's machine. This makes real credentials easily distinguishable from forged credentials: real credentials are on government-printed paper inside government-mailed envelopes which have stamped postage stamps – forged credentials are numbers on a screen. While it is theoretically possible for voters to forge credential envelopes, it is unrealistic to expect them to do so. So the practical problem of distributing credentials remains unsolved.

The authors of JCJ mostly refer to the registrar as a single, trusted entity. In fact, a few chapters of text are dedicated to explaining why the registrar needs to be a trusted entity. However, in appendix C they refer to the registrar as a group of multiple servers: "*the registration servers may if desired use designated verifier proofs to prove to each voter that the share they send is authentic*". So it is unclear if the registrar is a single trusted entity or a group of mutually distrusting entities.

Even if we assume public key infrastructure to support designated verifier proofs, and even if we assume that the practical issue of how to distribute credentials is somehow solved, we still have the fundamental issue of the registrar holding too much power: a single trusted entity acting as a registrar could copy voters' credentials to an adversary, who could then override their votes without the voters noticing.

Civitas' differences compared to JCJ

Civitas, based on JCJ, was first described in [20] by Clarkson et al. who also published a prototype implementation[71] accompanying the article. Civitas differs from JCJ in several ways. The following are the most important differences from our research perspective:

- Civitas introduces a distributed mechanism for vote storage. Instead of relying on a single trusted server for vote storage, mutually distrusting entities can set up virtual ballot boxes. A voter can submit their vote to some or all of the ballot boxes. The vote will be recorded unless all of the ballot boxes are misbehaving.
- Civitas introduces two additional keys: a registration key and a designation key (in addition to voter credentials). The registration key is used to authenticate the voter to the registrar. The designation key is needed for the designated verifier proof (as we discussed earlier, this was a missing piece in JCJ). The authors acknowledge the need for public key infrastructure in any real-world implementation of Civitas.

[71] https://www.cs.cornell.edu/projects/civitas/ (accessed 15.8.2019)

- Civitas implements n-out-of-n quorum for the registrar. This greatly reduces the potential consequences for having a corrupt or faulty registrar. As we noted earlier, a single-entity JCJ registrar could fabricate the entire tally. In comparison, the worst thing a single member of the registrar quorum in Civitas could do is refuse issuing credentials to valid voters. While this threat is orders of magnitude smaller, it is nonetheless a serious threat that needs to be mitigated. The authors acknowledge this and speculate that perhaps a mechanism could be developed to allow the voter to prove to third parties when a registration authority is misbehaving (currently, coercion-resistance properties prevent the voter from being able to prove this). Küsters and Truderung [51] describe an additional weakness in the n-out-of-n registrar quorum: a single corrupted member of the registrar quorum would be able to mount a forced abstention attack.

- Civitas implements a non threshold cryptosystem for tallying (requiring n-out-of-n quorum in order to produce the tally) instead of the threshold cryptosystem proposed in JCJ (requiring only k-out-of-n quorum). The authors do not justify this change, but presumably, a non threshold cryptosystem was easier to implement. This change gives a lot of power to each of the tallying authorities: any one of them can now prevent the counting of votes, whether it is due to honest mistakes (such as losing the private key) or corruption (such as politicians attempting to evoke a new election). It seems realistic that a corrupted authority could do this one time under the guise of technical issues and get away with it. We would like to highlight that the distributed nature of Civitas actually makes this threat *worse* compared to centralized schemes: in a centralized scheme only one authority can halt the election, in Civitas any one of several authorities can.

Civitas' quorum improvements

We present two quorum improvements which have been proposed to Civitas since the publication of the original article. One strengthens availability in the registration phase, the other in the tallying phase.

1. Registration: k-out-of-n quorum instead of n-out-of-n quorum, proposed by Shirazi et al. [89] The voter contacts a subset (possibly all) of quorum members to ask for credential shares, and after validating the responses, the voter commits to at least k shares. The main advantage of this change is that a single misbehaving quorum member will be unable to prevent a voter from registering. (The authors also discuss a different, inferior proposal where the voter has to commit to a subset of quorum members *before* contacting them.)

2. Tallying: k-out-of-n quorum instead of n-out-of-n quorum, proposed by Clarkson et al. [23] (one of the original authors of Civitas along with two new co-authors). The main advantage of this change is similar as above: a single misbehaving quorum member will be unable to prevent tallying. In order to achieve this, the authors replace Civitas' El Gamal cryptosystem with a threshold cryptosystem based on Shamir's Secret Sharing [88].

Civitas' credential handling improvements

While Civitas addressed some credential handling issues in JCJ, some severe issues remained unsolved. Neumann et al. address these issues from a very practical perspective and propose improvements to Civitas. Their scheme is not conclusively defined in a single article, but rather in two separate articles which should be considered together: [67] and [66].

Neumann et al. [67][66] propose the use of smart cards for credential handling. Initially, the voter sets up their smart card in a supervised environment at a registration authority's premises (so we can be confident that voters are not coerced during the registration phase). Later, when the voter is ready to send their vote, they need to input a pin on their smart card. If they are coerced, they input any invalid pin, which will be used to generate a forged credential, and if they are not coerced, they will input the real pin, which will be used to unlock the real credential so that the vote will be counted.

Note that these smart cards can not be used for any other purpose – otherwise distinguishing between real and forged pins would be trivial (for example, if a smart card was used to buy groceries online, the user would need to get feedback if the order succeeded or not, thus revealing whether the pin was real or forged). The high cost of single-purpose smart cards would be very difficult to justify politically (compared to general-purpose smart cards, such as those used in Estonia). Nonetheless, we consider single-purpose smart cards to be a realistic option, even if it is politically difficult to push.

Neumann et al. [67][66] also propose that the card reader would have an internal screen which could display the hash of the signed ballot. The voter could then use another device, like a smartphone app, to verify that the vote contains their choice. This ability would provide partial cast-as-intended verifiability (note that the voter still could not verify that they entered their pin correctly, so it does not provide complete cast-as-intended verifiability). In essence, this removes the need to trust voters' general-purpose home computers. If malware (or user error) lead to an incorrect choice on the ballot, the voter would detect this with the smartphone app and simply send another vote on a different device. Note that this still requires trust in the smart card and the smart card reader (and trust that *at most one* of the voter's devices are compromised).

However, Neumann et al. [67][66] do not explain how the forged credentials are generated. In particular, they do not address which credential shares are forged. Suppose there are 5 registration authorities and 1 of them is colluding with an adversary. If the smart card generates 5 forged credential shares, then the corrupted registration authority will be able to notice that the voter is forging credentials. This threat is partially mitigated by the quorum improvements proposed by Shirazi et al. [89] (which allow the voter to request credential shares only from a subset of registration authorities). However, this mitigation is insufficient: an adversary may coerce the voter to request credential shares from the corrupt registration authority, and furthermore, the voter may not know which authorities are corrupted.

In order to improve coercion-resistance, we propose an amendment to Neumann et al.'s [67][66] smart card scheme: the smart card should forge only 1 credential share. In the case of 1 out of 5 authorities being corrupted, this gives the voter an 80% chance of passing a forged credential without detection, which is not good, but is markedly better than the 0% chance the voter previously had.

The voter could be given the opportunity to select which registration authority's share will be forged. This would be useful in the event that the voter knows which authority is corrupted. However, it is imperative that this opportunity is only given during the supervised registration phase, and not in unsupervised conditions. Otherwise, a physically present coercer might be able to reveal the credential by coercing the user into forging two different credentials such that a different part is forged in each. This might be possible, for example, if the selection of forged credential shares becomes a configurable setting in the client application.

We would like to note that the amendment we proposed is not particularly novel: the original Civitas scheme already describe the voter as having the ability to choose which credential shares are forged. We merely applied the same idea into the smart card extension of Civitas and specified how this idea can be applied securely.

With the introduction of smart cards [67][66], Civitas becomes vulnerable to forced abstention attacks. An adversary can simply demand the voter to relinquish their smart card (physically, in-person). The voter will not have a copy of the smart card (and it would be prohibitively expensive to provide each voter with a random amount of identical smart cards). Civitas does not have a method for revoking and reissuing credentials. While this would be a minor issue in the original Civitas scheme with its easily copyable paper credentials, it is a major issue when the credentials are tied to smart cards. As far as we know, we are the first to identify this weakness.

It is not immediately clear how a revocation policy could be implemented without breaking coercion-resistance. An alternative approach to prevent these forced abstention attacks would be some kind of recovery code system. A voter could have several paper copies of these recovery codes hidden. If an adversary demands that they relinquish these recovery codes along with the smart card, they could simply relinquish some of the copies. An adversary has no way of discovering how many copies there are. The voter could later initialize a new smart card with the recovery codes. We regard this as future work.

Civitas and individual verifiability

During our literature review we found several descriptions of individual verifiability in Civitas [20][92][67]. All of them were either incorrect, misleading, or incomplete. Next we will walk through different properties of individual verifiability, explain how Civitas relates to these properties, and elaborate what is wrong with the descriptions in literature. As a reminder, a voting scheme is said to have individual verifiability if the voter can verify that their vote is cast as intended, recorded as cast, and count as recorded (see section 3 for a detailed explanation).

Cast-as-intended verification is disabled by design: although the user interface can be designed to verify the choice of voters, there is no verification for the validity of credentials. We expect that a significant proportion of voters will accidentally input forged credentials when they intend to use real credentials, and as a result, their vote will not be counted and there is no mechanism to enable voters to verify whether this has happened or not. In other words, voters can not verify that their vote has been counted, so Civitas does not provide individual verifiability.

In the smart card extension [67] [66] of Civitas, any PIN number will be accepted. As its authors acknowledge, there is a high risk of accidentally typing the incorrect PIN number even if the voter is thinking about the correct PIN number.

This risk is somewhat mitigated in the original version [20] of Civitas, which requires the voter to input actual credentials (instead of a PIN number which is used to generate a valid/forged credential in a smart card). Naturally, when the voter enters actual credentials instead of a PIN, it is very likely that a typo will be detected. However, with actual credentials, the voter needs to generate and store forged credentials in such a way that an adversary can not distinguish real credentials from forged credentials – the voter can not simply write ”real” next to the real credential. This naturally means that voters themselves will be at risk of forgetting which credential is real and which is forged.

In other words, we are not aware of a mechanism – in any variant of Civitas – that would prevent the voter from accidentally entering forged credentials. This blatantly obvious weakness appears to be ignored in the majority of articles in literature, and when it is noted, it is mentioned in passing only. For example, Spycher et al. [92] erroneously claim JCJ to have individual verifiability. The authors of Civitas [20] also erroneously claim that Civitas has individual verifiability (although they refer to this as *voter verifiability*, so there is some ambiguity regarding what they mean).

Neumann and Volkamer [67] also claim Civitas to have individual verifiability: *”Civitas ensures [...] end-to-end verifiability as composition of universal verifiability, i.e., any observer can verify that the votes stored in the ballot boxes are correctly tallied, and individual verifiability, i.e., each voter can verify that his vote has been cast as intended and stored as cast.”* Note how they claim Civitas to have cast-as-intended verifiability[72]. Also note how count-as-recorded verification is absent in their definition.

[72] Neumann and Volkamer did add an assumption that voters will never ”mistype or forget” their PIN numbers. It could be argued that given this assumption, Civitas has cast-as-intended verifiability. This may be technically correct. However, we would rather describe it as a cast-as-intended *assumption*, since we are merely *assuming* that voters cast their votes as intended, instead of *verifying* it. To be fair, the authors also discuss verifying the choice of the voter, so some aspects of cast-as-intended verifiability are provided, even though other aspects of cast-as-intended verifiability are not provided.

Recorded-as-cast verification is provided by the bulletin board. (The voter can verify that their vote appears on the bulletin board. See section 4 for more details.)

Count-as-recorded verification is comprised of several steps. Some of these steps can be verified by anyone, but some can be verified only by the voter. Next we will describe each of these steps, emphasizing who is able to perform each verification. We encourage readers to again look up figure 5 as a visualization aid to understand this process.

1. Proof of well-formedness. The voter must verify that their vote has survived this step. It is possible (depending on the variant of Civitas) that malware on the user's device has corrupted the vote in such a way that it will not pass this well-formedness check. Preferrably the voter should verify this (using a secondary device) before the voting period has ended, so that the voter has the opportunity to re-vote on a third device.

2. Elimination of votes with duplicate credentials. Proofs of correctness are posted to the bulletin board. Although outside auditors can be expected to verify these proofs, they are not sufficient to convince the voter that their vote has survived this step. It is possible that an adversary has gained access to the voter's credentials (for example, the spouse of the voter may have exploited physical access to the credentials). If the adversary has posted a vote with the same credentials *before* the voter, the voter's real vote will be eliminated, because only the first vote for each credential will survive this step[73]. This manipulation can not be caught by outside auditors and can only be verified by the voter themselves. As far as we know, we are the first to articulate the need for this verification.

3. Anonymization (mixing). The voter can either verify the proofs from this step or trust that outside auditors will verify the proof. If the proof is correct then the voter can trust that their vote survived the anonymization step.

[73] In some variants of Civitas the duplicate elimination step will keep the last vote instead of the first vote, but only if the vote contains certain additional proofs to demonstrate that the latter vote was posted by the same voter who posted the earlier vote. However, in our case the voter will be unaware of any previous vote and will be unable to post such proofs, so the first vote will still be counted and the latter vote will still be eliminated.

4. Elimination of votes with forged credentials. As with previous step, the talliers provide proof that this elimination is done correctly and the voter can either trust the auditors to verify this proof or verify it themselves. In addition, to convince the voter that their vote has survived this elimination, they need to be convinced that their credentials are valid. The voter was convinced of this during the registration step by use of designated verifier proofs.
5. Decryption of votes. As with previous step, the talliers provide proof that can be audited by anyone. The tally can then be verified by counting individual votes.

Naturally, these verifications would be done by client-side software. The voter would not have to do anything more complicated than click a button or scan a QR code.

JCJ/Civitas application-level denial-of-service attacks

The elimination of duplicate and forged credentials has quadratic time complexity in JCJ and Civitas. Smith [90] provides back-of-the-envelope calculations to demonstrate why this is unacceptable. As a quick example, if an adversary were to send a billion (10^9) votes – feasible with a single consumer PC – then the talliers would have to compute a billion billion (10^{18}) comparisons in order to eliminate duplicate credentials (not feasible). Koenig, Haenni and Fischli [48] named attacks of this nature *board flooding attacks*. Board flooding attacks are one example of *application-level denial-of-service attacks*, since they attack the availability of tallying.

Clarkson et al. reported experimental performance tests for their Civitas implementation and conclude that ”*cost, tabulation time, and security can be practical for real-world elections*” [20]. However, we argue that tabulation time for real-world elections would not be practical due to board flooding attacks. Despite acknowledging this vulnerability (”*Application-level denial of service is particularly problematic, because an adversary could insert [invalid votes] to inflate tabulation time*”), their performance tests limited invalid votes to 70% of total votes. If they had set realistic[74] limitations for invalid votes, their performance

[74] The use of percentages here may be confusing to readers (”70% of total votes”). As an example, suppose a country like Finland organizes an election for 5 million people. Suppose 3 million people vote. If invalid votes represent 70% of total votes, then there are 10 million total votes and 7 million of them are invalid. A single consumer PC would be able to generate 7 million invalid votes in less than a second. Clearly this is not a realistic upper bound for invalid votes. We do not attempt to calculate a realistic upper bound; we consider it obvious that quadratic time complexity is not acceptable in this case.

tests would have shown that the system grinds to a halt in a realistic setting.

Many authors have proposed various performance improvements in order to eliminate the pairwise verification and the consequent DoS vulnerability. The first of these proposals was described by Smith in [90], where he proposes the use of hash table lookups to reduce the time complexity of credential verification from quadratic to linear. In essence, Smith replaces the *pairwise* comparison of ciphertexts with a *global blind* comparison of ciphertexts. However, generating hashes from probabilistically encrypted credentials without compromising coercion resistance or verifiability is quite a challenge. The resulting scheme is complicated and has several weaknesses.

Weber et al. [101] discovered two issues in Smith's proposal and revised the scheme to account for them:

1. Hash collisions are possible in Smith's scheme. They may have the effect of turning valid votes invalid or invalid votes valid. The revised scheme takes hash collisions into account.

2. Weber et al. claim to have found a de-anonymization vulnerability in Smith's scheme that is related to timestamps, but they do not elaborate further. Smith's scheme relies on user-generated timestamps despite that users are not trusted and they do not share a global clock. Perhaps the vulnerability found by Weber et al. is somehow related to these issues. In any case, the revised scheme no longer relies on user-generated timestamps.

Clarkson et al. [19] discovered a serious vulnerability in Smith's scheme. In short, the encryption method used by Smith is not fit for this purpose. An adversary has a mathematical way of testing the validity of credentials, entirely breaking coercion-resistance. As noted by Araujo et al. [3], the same vulnerability also applies to the revised scheme proposed by Weber et al. in [101].

Araujo et al. [3] propose a new JCJ-based scheme to achieve the same performance improvement as Smith without losing any security properties. The key idea underlying Araujo et al.'s proposal is to entirely remove the comparison between submitted credentials and known-valid-credentials. Whereas JCJ used pairwise comparisons and Smith/Weber used global blind comparison, Araujo et al.'s scheme has no comparison at all. Instead, all information needed to validate the credential is sent along with the vote and can be validated by a quorum of talliers.

Spycher, Koenig and Haenni [92] note the following drawback in Araujo et al.'s approach: when a public voter roll no longer exists, ballot stuffing by a corrupted registrar becomes easier without detection. However, in our opinion this is only a minor drawback. In JCJ, as we noted earlier, a corrupted registrar already has the power to entirely fabricate the tally, so this change does not increase the registrar's power. In Civitas, this change does slightly increase the power of the registrar, but in Civitas the registrar consists of a quorum of mutually distrusting entities and the protection provided by the quorum construction is sufficient in our opinion. Nonetheless, it would be desirable if outside observers were able to verify that the voter credentials of supposedly-legitimate votes correspond to voter credentials in the voter roll. Spycher, Koenig and Haenni [92] propose a new scheme which has this slight advantage.

The idea behind Spycher, Koenig and Haenni's [92] proposal is quite simple: instead of comparing a vote's credential to *all* credentials in the voter roll (as in original JCJ and Civitas schemes), the voter identifies which entry in the voter roll corresponds to their credential and sends this information along with the vote.[75] This allows the elimination of forged credentials in linear time. In order to eliminate duplicate credentials, the authors use the method from Weber et al. [101] (which had a vulnerability in the forged-credential-elimination phase but not in the duplicate-credential-elimination phase).

In our opinion, both the proposal from Araujo et al. [3] and the proposal from Spycher, Koenig and Haenni [92] are viable solutions to bring Civitas' time complexity down from quadratic to linear. However, these proposals only address time complexity, not storage or networking costs. A board flooding attack may cause the public bulletin board in JCJ/Civitas to bloat to hundreds of terabytes in size. Even if a wealthy state can afford computing on a dataset of this size, normal people and small organizations can not. Thus, board flooding would severely undermine one of the key features of Civitas: verifiability. It would be preferrable if consumer grade PCs would be sufficient to verify the voting results. In order to achieve this, the size of the bulletin board must be somehow constrained from bloat by adversarial inputs.

Koenig, Haenni and Fischli propose such a scheme in [48]. Instead of allowing voters to generate a huge amount of forged credentials, each voter would be given a random amount of *dummy credentials*. The size of the bulletin board would be constrained by limiting the maximum amount of votes per voter to the amount of credentials given to them (the bulletin board would keep at most one vote per valid or dummy credential, and no votes per invalid credential). If a coercer

[75] In order to maintain confidentiality properties, some additional constructions are needed. We refer to [92] for details.

would demand a voter to give up all of their credentials, the voter could deceive the coercer by relinquishing all of the dummy credentials and lying that one of them is the valid credential. Due to the random amount of credentials, the coercer has no mechanism for distinguishing whether they received all of the credentials, or all except one.

Koenig, Haenni and Fischli describe several new vulnerabilities as a result of incorporating dummy credentials [48]. For example, a coercer might offer to pay for each additional credential relinquished by the voter, thus creating an incentive to relinquish the valid credential as well. In our opinion the vulnerabilities described by the authors would not be serious in practice (if the random distribution is chosen carefully). For example, if a voter is given 429 dummy credentials and 1 valid credential, and a coercer offers to pay for each credential relinquished, the monetary incentive for relinquishing the valid credential would be negligible. However, there is another issue: the authors describe their scheme in the context of JCJ and it does not appear to be easily transferrable to Civitas due to the distributed nature of ballot boxes in Civitas.

Clarification regarding which Civitas variant is in the comparison

We consider the following variant of Civitas: the original scheme [20] with the following improvements:

1. Credential handling improvements by Neumann et al. [67][66]
2. Quorum improvements by Shirazi et al. [89] and Clarkson et al. [23]

Note that we do not incorporate any improvements related to DoS resistance. This may be surprising, given that we discussed numerous promising options earlier. We would like to incorporate the dummy credential scheme proposed by Koenig, Haenni and Fischli [48], but their scheme was proposed as an extension of JCJ, not Civitas, and it may be incompatible with the distributed nature of Civitas' ballot boxes. The next best option to our knowledge would be the schemes by Araujo et al. [3] and Spycher, Koenig and Haenni [92]. However, as we discussed, their improvements are inadequate because they allow adversaries to bloat the bulletin board. We identify Civitas' DoS resistance as a promising research avenue where a practical solution may be found, possibly by building on the work of Koenig, Haenni and Fischli [48].

Security properties of remote e-voting with Civitas

P1. *Malware on voting device is unable to violate ballot secrecy.* Never holds. The voter inputs their choice as plaintext on a device which is susceptible to malware.

P2. *Malware on voting device is unable to manipulate votes.* Never holds due to inadequate dispute resolution, but considerably improved in this variant compared to the original Civitas scheme. The voter can use a secondary device, such as a smartphone, to verify the hash of their vote when it is displayed on the smart card (before it is transferred to the computer). If the computer is infected with malware (and the smart card is not), the voter can detect that their vote has been tampered with before any damage is done. The voter can then use a third device to vote. However, in the event of a large-scale malware campaign, the voter has no ability to prove that anything is wrong (to allow the courts to interfere). This means that only those voters who bother with the verification procedure would be able to detect the issue and protect their votes from manipulation. In this event the large-scale malware campaign would be able to manipulate a large amount of votes (from voters who do not bother with the verification).

P3. *Voter is able to keep their ballot as secret.* Almost always holds. In order to de-anonymize votes, k-out-of-n talliers would have to collude with each other and the registrars (the talliers would first collude with each other to link decrypted votes to credentials, and after that the talliers would collude with the registrars in order to link credentials to actual voters). Note that this property is slightly weakened in this variant compared to the original Civitas scheme (only k-out-of-n talliers needed to collude instead of n-out-of-n).

P4. *Voter is unable to prove to a large-scale vote-buyer how they voted.* Almost always holds (as long as the voter can identify at least one honest registrar). Although the voter can prove to vote-buyers that they voted a particular way and that their vote is recorded on the bulletin board, they can not prove that they used real credentials to cast this vote (they could have used forged credentials to defraud the vote-buyer). The voter can also relinquish their credentials to the vote-buyer so that they can vote on behalf of the voter, but the vote-buyer faces the same issue here: they can not verify whether the credentials are real or forged.

P5. *Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.* Almost always holds (as long as the voter can identify at least one honest registrar). Although the voter can relinquish copies of their credentials to vote-buyers, the voter is unable to prove that they have relinquished copies of the real credentials (the voter could have relinquished copies of forged credentials in order to deceive the vote-buyer).

P6. *Voter is unable to prove to their spouse how they voted.* Always holds. The spouse can physically watch the voter vote, but the voter can fool the spouse by entering an incorrect pin number on the smart card. The spouse has no ability to differentiate between a valid pin and an invalid pin. Note that this property is considerably improved in this variant compared to the original Civitas scheme. (As a reminder, we have defined the spouse as an adversary who can not collude with corrupt authorities.)

P7. *Voter is unable to prove to their spouse that they wasted their right to vote.* Never holds. The spouse can simply demand the voter to relinquish their smart card and thus prevent the voter from voting. No method exists to revoke and reissue credentials. Note that this property is actually weaker in this variant compared to the original Civitas scheme where the voter can store an arbitrary amount of paper copies of real and forged credentials (however, as we noted earlier, the original scheme has other, more severe issues).

P8. *Voter can ensure their ballot is not accidentally spoiled.* Never holds. Although the client side software can ensure that the ballot is of proper form, nothing prevents the voter from spoiling the ballot by accidentally inputting an incorrect pin on the smart card.

P9. *Voter can ensure their vote is recorded as cast.* Always holds. The voter can look up their vote on the public bulletin board. If the voter can not find their vote there, they can re-vote. If the bulletin board is discriminating against some voters by refusing to accept their votes, the voters can prove this with the help of the digital ballot boxes (which are run by different organizations).

P10. *Voter can detect if their vote is displaced (deleted, replaced or pre-empted).* Always holds. If a vote is deleted from the public bulletin board, organizations which monitor the bulletin board for changes would notice this. A vote can not be replaced in this variant of Civitas (in this variant, when multiple votes are made with the same credentials, only the first vote counts). A vote can, however, be pre-empted. For example, the spouse of the voter might exploit physical access to the credentials and secretly vote on behalf of the voter before the voter casts their real vote. The voter can detect this by verifying that their vote has survived the duplicate elimination -step.

P11. *The tally is counted correctly from recorded votes.* Always holds. Even if all of the talliers are misbehaving, they can not forge convincing proofs of the mixnet shuffling and decryption processes.

P12. *No ballot stuffing.* Almost always holds. Votes sent with invalid credentials are pruned during the ballot well-formedness check. Votes sent with forged credentials are pruned after anonymizing votes. All votes which survive this pruning correspond to credentials of eligible voters. If all of the registrars collude with each other, they can steal credentials and add votes to voters who did not actually vote.

P13. *Denial-of-service resistance.* Never holds. An adversary with a single consumer PC would be able to create a modest amount of invalid votes, which would prevent the tallying process from completing.

6 Comparison

In this section we present a comparison table of our results, guidance for interpretation, key takeaways from the comparison, and outlines for future work.

6.1 Comparison table

		Malware		Vote-buying, coercion					Verifiability, dispute res.					DoS
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13
In-person paper voting in	Finland	🟩	🟩	🟩	▨	▨	🟥	🟥	🟥	🟩	🟨	🟨	🟨	🟨
In-person paper voting with	Floating Receipts	▨	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟩	🟩	🟨	🟨
In-person paper voting with	Prêt à Voter	▨	🟩	▨	🟨	🟨	🟥	🟥	🟥	🟩	🟩	🟩	🟨	🟨
Remote paper voting in	Switzerland	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟨	🟨	🟨	🟨
Remote e-voting in	Switzerland	🟥	🟨	🟨	🟥	🟥	🟥	🟥	🟩	🟨	🟨	🟨	🟨	🟨
Remote e-voting in	Australia	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟩	🟨	🟨	🟨	🟨	🟨
Remote e-voting in	Estonia	🟥	🟥	🟨	🟨	🟨	🟩	🟩	🟩	🟩	🟥	🟨	🟨	🟨
Remote e-voting with	Helios	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟩	🟥	🟩	🟩	🟩	🟥
Remote e-voting with	Civitas	🟥	🟥	▨	▨	▨	🟩	🟥	🟥	🟩	🟩	🟩	▨	🟥

Color descriptions

🟥	**Never** holds, scheme does not provide this property.
🟨	Holds when **none** of the authorities are corrupted.
▨	Holds even if (any) **one** authority is corrupted.
🟩	Holds even if **all** authorities are corrupted.

Property descriptions (details in section 3.4)

P1	Malware on voting device is unable to violate ballot secrecy.
P2	Malware on voting device is unable to manipulate votes.
P3	Voter is able to keep their ballot as secret.
P4	Voter is unable to prove to a large-scale vote-buyer how they voted.
P5	Voter is unable to prove to a large-scale vote-buyer that they wasted their right to vote.
P6	Voter is unable to prove to their spouse how they voted.
P7	Voter is unable to prove to their spouse that they wasted their right to vote.
P8	Voter can ensure their ballot is not accidentally spoiled.
P9	Voter can ensure their vote is recorded as cast.
P10	Voter can detect if their vote is displaced (deleted, replaced or pre-empted).
P11	The tally is counted correctly from recorded votes.
P12	No ballot stuffing.
P13	Denial-of-service resistance.

6.2 Guidance for interpreting results

We would like to remind readers that the names of properties in the comparison table are just names; more precise descriptions are available in section 3.4. Readers who are wondering how we have arrived at a particular conclusion (”why is this red?”) will find a justification for every single claim by clicking on the name of the scheme in question. We hope that these descriptions will help readers confirm or dispute our findings more easily.

Readers should avoid the temptation of mentally ”scoring” schemes against each other based on the number of properties satisfied. The properties in the comparison are not equally important – for example, a violation of P7 is considered to be harmless by some authors [5], whereas a violation of P11 means that votes have been counted incorrectly. In addition, the value of properties depends on the use case. For example, Helios is a great choice for low-stakes elections, even though it enables vote-buying and coercion. High-stakes elections have clearly different requirements.

A common talking point in discussions around e-voting is that computer security can never be guaranteed, and therefore e-voting should never be used. Although there is a kernel of truth to that statement, we find it unreasonable to set such a high bar. We do not have a perfect voting system (paper or otherwise). Comparisons should be made between realistic options, not against idealized non-existing options.

For example, if we are discussing remote e-voting for *absent* voters, then the relevant comparison point should be whichever voting scheme absent voters are currently using – typically a remote paper voting scheme. If we are discussing remote e-voting for *all* voters, then the relevant comparison point should be whichever voting scheme is available to most voters – typically an in-person voting scheme. We provide advice for policymakers in appendix A.

Aspects not illustrated in the comparison table

The ”authority” abstraction is rather coarse. In some cases manipulation is possible by a single rogue employee, in other cases the collusion of multiple employees is required. In addition, consider the potential scale for manipulation. In e-voting schemes the effort required to manipulate a small amount of votes is typically the same as the effort required to manipulate a large amount of votes [102][8]. The same does not hold true for typical paper voting schemes; a corrupted postal worker would find it easy to manipulate a small amount of votes, but difficult to manipulate a large amount of votes (the postal worker may not be able to modify votes, but they would be able to selectively destroy them). These aspects are not in any way illustrated by our comparison.

The ”voting device” abstraction is also coarse. Certain schemes utilize consumer PCs (which are commonly infected with malware) whereas other schemes utilize more trustworthy devices, such as computers maintained by election officials for the sole purpose of voting.

To clarify, we believe the abstractions we chose are appropriate and provide the simplifications we need to present results in a concise manner. It is simply good to remember which aspects are inadequately represented in the comparison.

In addition, as we explained in section 2, it is good to remember that this thesis covers only security aspects of voting schemes. We do not cover other aspects of voting schemes, such as usability or understandability. Also, a voting scheme represents only part of a voting system (though arguably the most crucial part). Aspects such as cost estimation, adaption of best practices in software development, or operational security controls are crucial to voting systems, but they are not covered in this thesis.

Regarding unfair aspects of our comparison

As we noted in section 2, our intent was to provide an apples-to-apples comparison of voting schemes. In some aspects we fell short of that goal, and we want to be transparent about those issues.

Firstly, with regards to comparisons between existing real-world voting schemes and non-existing theorized voting schemes, there is an inherent unfairness in such comparisons. Although we could anticipate that real-world schemes have to face some challenges that theorized schemes may ignore, we were surprised by the extent of the problem. Furthermore, we did very little to alleviate this inherent unfairness in the comparison. We refer interested readers to the STAR-vote article [5], which describes practical limitations in designing a real-world voting scheme.

Secondly, we did not spend equal time researching different schemes. Some schemes received a lot of attention, other schemes received very little. In some cases, such as the Swiss postal voting scheme (section 5.4), there simply was not much literature available. In other cases, such as Civitas (section 5.9), we were really fascinated by the scheme and ended up reading a huge number of articles out of curiosity. By spending a lot of time on Civitas we were more likely to discover vulnerabilities, but also more likely to select a good variant in the comparison. Hopefully these effects cancel each other out at least to some extent.

The amount of variants in general turned out to be an issue. Before our literature search we were expecting that each voting scheme would be clearly described somewhere, perhaps on a website or perhaps in a scientific article by its authors. Surprisingly, this was not the case. In fact, no voting scheme in our comparison is explicitly defined in a single location. Oftentimes the first publication of a scheme is in some ways broken, and a multitude of follow-up articles is written in an effort to improve the scheme (for example, JCJ/Civitas). One might visualize these articles as a tree where different branches represent different variants. In some cases the scheme is not completely defined anywhere (not even by combining information from different sources). In these cases we had to ”fill the blanks” by guessing (for example, Finland, Floating Receipts, and both Swiss schemes). We made a huge effort to select the best variant of each scheme, but naturally, spending more time on one scheme is more likely to yield a better variant, so we have been treating schemes unfairly in this sense.

6.3 Key takeaways from the comparison

In this section we discuss various observations from the comparison table. We also relate these observations to our research questions whenever applicable. As a reminder, we articulated the following research questions in section 2.1:

RQ 1. *What strengths and weaknesses do different schemes have relative to each other?*

RQ 2. *Are some of the weaknesses a result of unavoidable tradeoffs?*

RQ 3. *Which voting schemes are most suitable for different use cases?*

Starting with the obvious, remote e-voting schemes tend to be vulnerable to client-side malware (P1, P2). Even though code voting and verification features are commonly incorporated, they are often utilized inadequately. We were surprised that none of the remote e-voting schemes in our review were safe from client-side malware. In reference to RQ2, we would characterize this as a tradeoff between security (paper) and convenience (e-voting), although we wouldn’t characterize it as an *unavoidable* tradeoff.

Availability (P13) turned out to be an issue for two academic schemes in our review: Helios and Civitas. In the case of Helios there appears to be a tradeoff between denial-of-service resistance and ballot stuffing resistance; the DoS vulnerability could be fixed at the expense of introducing a ballot stuffing vulnerability (RQ2). In the case of Civitas, we reviewed a multitude of proposals for improved availability, but did not find an adequate solution. Civitas does not appear to be practically viable (RQ1).

Civitas provides – by far – the most extensive protection against corrupt authorities (RQ3). Civitas attempts to provide coercion-resistant voting, but none of the variants we looked at fully redeemed that promise. In the end we settled with a variant where only P7 is violated (the least important of our confidentiality properties) (RQ1). Civitas appears to make an unavoidable tradeoff between coercion-resistance properties and cast-as-intended verifiability (P8). In short, the ability to fool coercers by typing an incorrect pin number on a smart card opens up the possibility to *accidentally* typing an incorrect pin number (RQ2). With regards to the other integrity properties (P9-P12), Civitas is the only scheme in our review which provides all of them without a trusted authority (RQ1).

Switzerland has a long history of widespread postal voting. In the last decade they have made great strides towards remote e-voting. Critics of this movement are opposing remote e-voting due to security concerns, while proponents are claiming it to be just as secure as postal voting. Turns out they are both right. The Swiss remote e-voting scheme is horribly insecure and their remote paper voting scheme is just as bad, with only minor differences. Both schemes are highly vulnerable to vote-buying and coercion and integrity of the results depends mainly on trusted authorities. The Australian scheme is developed by the same vendor (Scytl), which may explain similarities.

Estonia's scheme is the only one in our review which protects voters from spousal coercion and similar adversaries (RQ3). This comes at a cost of disabling one crucial verifiability (P10), but the scheme relies heavily on trusted authorities in any case, so this tradeoff does not actually weaken the scheme.

We were surprised to discover how well Floating Receipts fared in comparison to all the other schemes. It was the only scheme which was able to satisfy all integrity properties (although requiring trust in authorities in case of P12, ballot stuffing). Given the simplicity of the scheme and the highly desirable security properties, we are wondering why Floating Receipts has not received more attention. (RQ1, RQ3)

6.4 Future work

During this thesis we outlined several potential topics for future research. However, we thought it was best to leave these in context[76] rather than attempt to describe them in this section.

If you wish to cite this work, we encourage linking to `https://www.attejuvonen.fi/thesis/`, which contains both historical and latest versions of this thesis. We kindly ask that readers send us corrections for any errors they spot. Evaluating all of the security properties for all of the schemes was an excruciatingly laborious process and we fear that we may have missed some errors despite all our diligence.

We hope that researchers will find our framework useful in evaluation of new voting schemes. We also encourage adaptations and imitations of our framework. We hereby license this thesis permissively under *Creative Commons Attribution 4.0 International*.[77]

Acknowledgements

My sincerest gratitude goes out to everyone who supported me on this long journey. My supervisor Valtteri Niemi provided invaluable feedback on both literary aspects and technical content. Valtteri called me out on many dubious claims in early drafts, leading to better argumentation and referencing, more accurate claims, and in some cases to the discovery of critical errors.

My employer Futurice provided me incredible support to graduate. Futurice gave me a year off work to study, organized a thesis writing camp, and supported this thesis financially. How amazing is it that Futurice – a digital consultancy unrelated to voting schemes – supports my thesis to this extent despite not receiving any direct benefit from it? I also want to thank Jesse Haapoja, the instructor of our thesis writing camp, for providing helpful guidance and literary feedback. The pressure from the thesis writing camp was certainly a contributing factor to finishing this thesis in time.

Any errors still remaining in this thesis should be attributed to me and not to any of the wonderful people who provided helpful feedback. Finally, I want to thank my wife Marianne for all her love and understanding throughout this exhausting writing process.

[76] You can locate these remarks by searching for ”future”.

[77] https://creativecommons.org/licenses/by/4.0/ (accessed on 27.9.2019)

References

[1] Adida, Ben: *Advances in cryptographic voting systems.* 2006.

[2] Adida, Ben: *Helios: Web-based open-audit voting.* In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[3] Araujo, Roberto, Foulle, Sébastien, and Traoré, Jacques: *A practical and secure coercion-resistant scheme for internet voting.* In *Towards Trustworthy Elections*, pages 330–342. Springer, 2010.

[4] Bannet, Jonathan, Price, David W, Rudys, Algis, Singer, Justin, and Wallach, Dan S: *Hack-a-vote: Security issues with electronic voting systems.* IEEE Security & Privacy, 2(1):32–37, 2004.

[5] Bell, Susan, Benaloh, Josh, Byrne, Michael D, DeBeauvoir, Dana, Eakin, Bryce, Kortum, Philip, McBurnett, Neal, Pereira, Olivier, Stark, Philip B, Wallach, Dan S, *et al.*: *Star-vote: A secure, transparent, auditable, and reliable voting system.* In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[6] Benaloh, Josh and Tuinstra, Dwight: *Receipt-free secret-ballot elections.* In *STOC*, volume 94, pages 544–553, 1994.

[7] Benaloh, Josh Daniel Cohen: *Verifiable secret-ballot elections.* 1989.

[8] Bernhard, Matthew, Benaloh, Josh, Halderman, J Alex, Rivest, Ronald L, Ryan, Peter YA, Stark, Philip B, Teague, Vanessa, Vora, Poorvi L, and Wallach, Dan S: *Public evidence from secret ballots.* In *International Joint Conference on Electronic Voting*, pages 84–109. Springer, 2017.

[9] Boneh, Dan: *The decision diffie-hellman problem.* In *International Algorithmic Number Theory Symposium*, pages 48–63. Springer, 1998.

[10] Bretschneider, Jennie, Flaherty, Sean, Goodman, Susannah, Halvorson, Mark, Johnston, Roger, Lindeman, Mark, Rivest, Ronald L, Smith, Pam, and Stark, Philip B: *Risk-limiting post-election audits: Why and how.* California, Estados Unidos: Risk-Limiting Audits Working Group, 2012.

[11] Brightwell, Ian, Cucurull, Jordi, Galindo, David, and Guasch, Sandra: *An overview of the ivote 2015 voting system.* New South

Wales Electoral Commission, Australia, Scytl Secure Electronic Voting, Spain, 2015.

[12] Brusco, Valeria, Nazareno, Marcelo, and Stokes, Susan Carol: *Vote buying in argentina.* Latin American Research Review, 39(2):66–88, 2004.

[13] Callen, Michael and Long, James D: *Institutional corruption and election fraud: Evidence from a field experiment in afghanistan.* American Economic Review, 105(1):354–81, 2015.

[14] Cardillo, Anthony and Essex, Aleksander: *The threat of ssl/tls stripping to online voting.* In *International Joint Conference on Electronic Voting*, pages 35–50. Springer, 2018.

[15] Chaum, David: *Zero-knowledge undeniable signatures.* In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 458–464. Springer, 1990.

[16] Chaum, David: *Secret-ballot receipts: True voter-verifiable elections.* IEEE security & privacy, 2(1):38–47, 2004.

[17] Chaum, David, Ryan, Peter YA, and Schneider, Steve: *A practical voter-verifiable election scheme.* In *European Symposium on Research in Computer Security*, pages 118–139. Springer, 2005.

[18] Chaum, David L: *Untraceable electronic mail, return addresses, and digital pseudonyms.* Communications of the ACM, 24(2):84–90, 1981.

[19] Clarkson, Michael, Chong, Stephen, and Myers, Andrew: *Civitas: A secure remote voting system.* In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[20] Clarkson, Michael R, Chong, Stephen, and Myers, Andrew C: *Civitas: Toward a secure voting system.* In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368. IEEE, 2008.

[21] Culnane, Chris, Ryan, Peter YA, Schneider, Steve, and Teague, Vanessa: *vvote: a verifiable voting system.* ACM Transactions on Information and System Security (TISSEC), 18(1):3, 2015.

[22] Culnane, Chris, Ryan, Peter YA, Schneider, Steve, and Teague, Vanessa: *vvote: a verifiable voting system.* ACM Transactions on Information and System Security (TISSEC), 18(1):3, 2015.

[23] Davis, Adam M, Chmelev, Dmitri, and Clarkson, Michael R: *Civitas: Implementation of a threshold cryptosystem.* 2008.

[24] Desmedt, Yvo and Frankel, Yair: *Threshold cryptosystems.* In *Conference on the Theory and Application of Cryptology*, pages 307–315. Springer, 1989.

[25] Eldridge, Mark: *A trustworthy electronic voting system for australian federal elections.* arXiv preprint arXiv:1805.02202, 2018.

[26] Esteve, Jordi Barrati, Goldsmith, Ben, and Turner, John: *International experience with e-voting.* International Foundation for Electoral Systems, 2012.

[27] Fiat, Amos and Shamir, Adi: *How to prove yourself: Practical solutions to identification and signature problems.* In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer, 1986.

[28] Fischlin, Marc: *Trapdoor commitment schemes and their applications.* PhD thesis, Citeseer, 2001.

[29] Fouard, Laure, Duclos, Mathilde, and Lafourcade, Pascal: *Survey on electronic voting schemes.* supported by the ANR project AVOTÉ, 2007.

[30] Fujisaki, Eiichiro and Okamoto, Tatsuaki: *Statistical zero knowledge protocols to prove modular polynomial relations.* In *Annual International Cryptology Conference*, pages 16–30. Springer, 1997.

[31] Furukawa, Jun and Sako, Kazue: *An efficient scheme for proving a shuffle.* In *Annual International Cryptology Conference*, pages 368–387. Springer, 2001.

[32] Goggin, Stephen N, Byrne, Michael D, and Gilbert, Juan E: *Post-election auditing: effects of procedure and ballot type on manual counting accuracy, efficiency, and auditor satisfaction and confidence.* Election Law Journal: Rules, Politics, and Policy, 11(1):36–51, 2012.

[33] Goldwasser, Shafi, Micali, Silvio, and Rackoff, Charles: *The knowledge complexity of interactive proof systems.* SIAM Journal on computing, 18(1):186–208, 1989.

[34] Golle, Philippe, Jakobsson, Markus, Juels, Ari, and Syverson, Paul: *Universal re-encryption for mixnets.* In *Cryptographers' Track at the RSA Conference*, pages 163–178. Springer, 2004.

[35] Halderman, J Alex and Teague, Vanessa: *The new south wales ivote system: Security failures and verification flaws in a live online election.* In *International conference on e-voting and identity*, pages 35–53. Springer, 2015.

[36] Harvey, Cole J: *Changes in the menu of manipulation: Electoral fraud, ballot stuffing, and voter pressure in the 2011 russian election.* Electoral studies, 41:105–117, 2016.

[37] Heiberg, Sven, Martens, Tarvi, Vinkel, Priit, and Willemson, Jan: *Improving the verifiability of the estonian internet voting scheme.* In *International Joint Conference on Electronic Voting*, pages 92–107. Springer, 2016.

[38] Heiberg, Sven and Willemson, Jan: *Verifiable internet voting in estonia.* In *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, pages 1–8. IEEE, 2014.

[39] Hirt, Martin and Sako, Kazue: *Efficient receipt-free voting based on homomorphic encryption.* In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 539–556. Springer, 2000.

[40] Horwitz, Daniel A: *A picture's worth a thousand words: Why ballot selfies are protected by the first amendment.* SMU Sci. & Tech. L. Rev., 18:247, 2015.

[41] Inguva, Srinivas, Rescorla, Eric, Shacham, Hovav, and Wallach, Dan S: *Source code review of the hart intercivic voting system.* University of California, Berkeley under contract to the California Secretary of State, 2007.

[42] Jakobsson, Markus and Juels, Ari: *Mix and match: Secure function evaluation via ciphertexts.* In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–177. Springer, 2000.

[43] Jakobsson, Markus, Juels, Ari, and Rivest, Ronald L: *Making mix nets robust for electronic voting by randomized partial checking.* In *USENIX security symposium*, pages 339–353. San Francisco, USA, 2002.

[44] Jakobsson, Markus, Sako, Kazue, and Impagliazzo, Russell: *Designated verifier proofs and their applications.* In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 143–154. Springer, 1996.

[45] Juels, Ari, Catalano, Dario, and Jakobsson, Markus: *Coercion-resistant electronic elections.* In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70. ACM, 2005.

[46] Juels, Ari, Catalano, Dario, and Jakobsson, Markus: *Coercion-resistant electronic elections.* In *Towards Trustworthy Elections*, pages 37–63. Springer, 2010.

[47] Kiniry, Joseph R., Zimmerman, Daniel M., Wagner, Daniel, Robinson, Philip, Foltzer, Adam, and Morina, Shpatar: *The future of voting: End-to-end verifiable internet voting - specification and feasibility study.* 2015. `https://www.usvotefoundation.org/E2E-VIV`.

[48] Koenig, Reto, Haenni, Rolf, and Fischli, Stephan: *Preventing board flooding attacks in coercion-resistant electronic voting schemes.* In *IFIP International Information Security Conference*, pages 116–127. Springer, 2011.

[49] Kohno, Tadayoshi, Stubblefield, Adam, Rubin, Aviel D, and Wallach, Dan S: *Analysis of an electronic voting system.* In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 27–40. IEEE, 2004.

[50] Küsters, Ralf and Müller, Johannes: *Cryptographic security analysis of e-voting systems: Achievements, misconceptions, and limitations.* In *International Joint Conference on Electronic Voting*, pages 21–41. Springer, 2017.

[51] Küsters, Ralf and Truderung, Tomasz: *An epistemic approach to coercion-resistance for electronic voting protocols.* In *2009 30th IEEE Symposium on Security and Privacy*, pages 251–266. IEEE, 2009.

[52] Küsters, Ralf, Truderung, Tomasz, and Vogt, Andreas: *Verifiability, privacy, and coercion-resistance: New insights from a case study.* In *2011 IEEE Symposium on Security and Privacy*, pages 538–553. IEEE, 2011.

[53] Küsters, Ralf, Truderung, Tomasz, and Vogt, Andreas: *Clash attacks on the verifiability of e-voting systems.* In *2012 IEEE Symposium on Security and Privacy*, pages 395–409. IEEE, 2012.

[54] Lewis, Sarah Jamie, Pereira, Olivier, and Teague, Vanessa: *Addendum to how not to prove your election outcome.* 2019.

[55] Lewis, Sarah Jamie, Pereira, Olivier, and Teague, Vanessa: *Ceci n'est pas une preuve - the use of trapdoor commitments in bayer-groth proofsand the implications for the verifiabilty of thescytl-swisspost internet voting system*, 2019.

[56] Lewis, Sarah Jamie, Pereira, Olivier, and Teague, Vanessa: *How not to prove your election outcome.* 2019.

[57] Li, Huian, Kankanala, Abhishek Reddy, and Zou, Xukai: *A taxonomy and comparison of remote voting schemes.* In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2014.

[58] Lindeman, Mark and Stark, Philip B: *A gentle introduction to risk-limiting audits.* IEEE Security & Privacy, 10(5):42–49, 2012.

[59] Locher, Philipp, Haenni, Rolf, and Koenig, Reto E: *Analysis of the cryptographic implementation of the swiss post voting protocol.* 2019.

[60] Lowry, Svetlana Z and Vora, Poorvi L: *Desirable properties of voting systems.* In *NIST E2E workshop*, 2009.

[61] Mercuri, Rebecca T: *Physical verifiability of computer systems.* In *5th International Computer Virus and Security Conference.* Citeseer, 1992.

[62] Meter, Christian: *Design of distributed voting systems.* arXiv preprint arXiv:1702.02566, 2017.

[63] Moran, Tal and Naor, Moni: *Receipt-free universally-verifiable voting with everlasting privacy.* In *Annual International Cryptology Conference*, pages 373–392. Springer, 2006.

[64] Mursi, Mona FM, Assassa, Ghazy MR, Abdelhafez, Ahmed, and Samra, Kareem M Abo: *On the development of electronic voting: a survey.* International Journal of Computer Applications, 61(16), 2013.

[65] Neff, C Andrew: *A verifiable secret shuffle and its application to e-voting.* In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[66] Neumann, Stephan, Feier, Christian, Volkamer, Melanie, and Koenig, Reto: *Towards a practical jcj/civitas implementation.* INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt, 2013.

[67] Neumann, Stephan and Volkamer, Melanie: *Civitas and the real world: problems and solutions from a practical point of view.* In *2012 Seventh International Conference on Availability, Reliability and Security*, pages 180–185. IEEE, 2012.

[68] Niemi, Valtteri and Renvall, Ari: *How to prevent buying of votes in computer elections.* In *International Conference on the Theory and Application of Cryptology*, pages 164–170. Springer, 1994.

[69] Norden, Lawrence D and Famighetti, Christopher: *America's Voting Machines at Risk.* Brennan Center for Justice at New York University School of Law, 2015.

[70] Oikeusministeriö: *Presidentinvaali 2018 vaaliohjeet nro 2: Vaalilautakunnan tehtävät.*

[71] Oikeusministeriö: *Presidentinvaali 2018 vaaliohjeet nro 4: Ennakkoäänestys kotimaan yleisessä ennakkoäänestyspaikassa.*

[72] Oikeusministeriö: *Presidentinvaali 2018 vaaliohjeet nro 8: Vaalipiirilautakunnan tehtävät.*

[73] Pieters, Wolter: *Verifiability of electronic voting: between confidence and trust.* In *Data Protection in a Profiled World*, pages 157–175. Springer, 2010.

[74] Puiggalí, Jordi and Rodríguez-Pérez, Adrià: *Defining a national framework for online voting and meeting its requirements: the swiss experience.* E-Vote-ID 2018, page 82, 2018.

[75] Raimo Ahola, Helsingin vaalipiirilautakunta. Private communication.

[76] Randell, Brian and Ryan, Peter YA: *Voting technologies and trust.* IEEE Security & Privacy, 4(5):50–56, 2006.

[77] Reed, Michael G, Syverson, Paul F, and Goldschlag, David M: *Anonymous connections and onion routing.* IEEE Journal on Selected areas in Communications, 16(4):482–494, 1998.

[78] Riemann, Robert: *Towards Trustworthy Online Voting: Distributed Aggregation of Confidential Data.* PhD thesis, Université de Lyon, 2017.

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

[80] Rivest, Ronald L and Smith, Warren D: *Three voting protocols: Threeballot, vav, and twin.* USENIX/ACCURATE Electronic Voting Technology (EVT 2007), 2007.

[81] Rjašková, Zuzana: *Electronic voting schemes.* Diplomová práca, Bratislava, 2002.

[82] Ryan, Peter and Peacock, Thea: *Prêt à Voter: a system perspective.* University of Newcastle upon Tyne, 2005.

[83] Ryan, Peter YA and Schneider, Steve A: *Prêt à voter with re-encryption mixes.* In *European Symposium on Research in Computer Security*, pages 313–326. Springer, 2006.

[84] Ryan, Peter YA and Teague, Vanessa: *Pretty good democracy.* In *International Workshop on Security Protocols*, pages 111–130. Springer, 2009.

[85] Sampigethaya, Krishna and Poovendran, Radha: *A framework and taxonomy for comparison of electronic voting schemes.* Computers & Security, 25(2):137–153, 2006.

[86] Scytl: *Scytl svote complete verifiability security proof report document 1.0.* `https://www.post.ch/en/business-solutions/e-voting/publications-and-source-code`, visited on 12.8.2019.

[87] Scytl: *Swiss online voting system cryptographic proof of individual verifiability.* `https://www.post.ch/en/business-solutions/e-voting/publications-and-source-code`, visited on 12.8.2019.

[88] Shamir, Adi: *How to share a secret.* Communications of the ACM, 22(11):612–613, 1979.

[89] Shirazi, Fateme, Neumann, Stephan, Ciolacu, Ines, and Volkamer, Melanie: *Robust electronic voting: Introducing robustness in civitas.* In *2011 International Workshop on*

Requirements Engineering for Electronic Voting Systems, pages 47–55. IEEE, 2011.

[90] Smith, Warren D: *New cryptographic election protocol with best-known theoretical properties.* In *Proc. of Workshop on Frontiers in Electronic Elections*, pages 1–14, 2005.

[91] Springall, Drew, Finkenauer, Travis, Durumeric, Zakir, Kitcat, Jason, Hursti, Harri, MacAlpine, Margaret, and Halderman, J Alex: *Security analysis of the estonian internet voting system.* In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715. ACM, 2014.

[92] Spycher, Oliver, Koenig, Reto, Haenni, Rolf, and Schläpfer, Michael: *A new approach towards coercion-resistant remote e-voting in linear time.* In *International Conference on Financial Cryptography and Data Security*, pages 182–189. Springer, 2011.

[93] Stark, Philip B and Wagner, David: *Evidence-based elections.* IEEE Security & Privacy, 10(5):33–41, 2012.

[94] State Electoral Office of Estonia: *General framework of electronic voting and implementations thereof at national elections in estonia*, 2017. `https://www.valimised.ee/sites/default/files/uploads/eng/IVXV-UK-1.0-eng.pdf`, visited on 31.7.2019.

[95] Stefanelli, Raffaele and Monnat, Denis Moreland Xavier: *A secure e-voting infrastructure. implementation by swiss post.* Second In, page 326, 2017.

[96] Strauss, Charlie EM: *A critical review of the triple ballot voting system, part 2: Cracking the triple ballot encryption.* Unpublished draft, http://cems. browndogs. org/pub/voting/tripletrouble. pdf, 74, 2006.

[97] Swiss Post: *Post e-voting: system documentation.* `https://www.post.ch/en/business-solutions/e-voting/publications-and-source-code`, visited on 12.8.2019.

[98] UN General Assembly: *Universal declaration of human rights.* 302(2), 1948.

[99] United Nations Office on Drugs and Crime: *Mechanism for the review of implementation of the united nations convention against corruption - state under review: Finland.*

[100] Verbij, Ruud Paul: *Dutch e-voting opportunities. risk assessment framework based on attacker resources.* Master's thesis, University of Twente, 2014.

[101] Weber, Stefan G, Araujo, Roberto, and Buchmann, Johannes: *On coercion-resistant electronic elections with linear work.* In *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 908–916. IEEE, 2007.

[102] Yi, Xun and Okamoto, Eiji: *Practical remote end-to-end voting scheme.* In *International Conference on Electronic Government and the Information Systems Perspective*, pages 386–400. Springer, 2011.

> “*Best practices for Internet voting are like best practices for drunk driving.*”
>
> – Ronald Rivest

A Opinionated advice for policymakers

After looking at the comparison table with schemes like Helios and Civitas, it may seem like remote e-voting isn't that bad. It is! This thesis considered the security of voting *schemes*; not other security aspects, such as those related to implementation, development, or procurement. If you are wondering should your government procure a contract to develop a remote e-voting system, the answer is almost invariably no. Even if it is theoretically possible to procure, develop and implement a remote e-voting system properly, no government to date has been able to do that.[78]

Instead of pursuing remote e-voting, governments should pursue advancements in supervised in-person voting. Cryptographic in-person voting systems have been experimented in real-world elections with promising results. For example, Prêt à Voter was implemented in Australia [22] and STAR-vote was implemented in the United States [5]. These case studies highlight many practical concerns which are often ignored in academic literature.

Another option is incremental improvements on top of the existing system. For example, the Finnish voting system would benefit from cast-as-intended verification. Finland could also introduce compliance and risk-limiting audits [93] on top of the current system.

In addition, I would like to remind policymakers of the horrid track record that government certification programs have with regards to voting equipment (section 1.4) and verifiability requirements (section 5.5). These certification programs stifle competition and technological progress without providing tangible security benefits [79][93][56]. I recommend that policymakers do not attempt to define every low-level minutiae details of a voting system, but instead seek to provide high-level requirements for evidence-based elections.

[78] As we noted in section 1, we made an effort to research remote e-voting systems in the wild and discovered only 3 systems which were (in our opinion) impactful: Australia, Estonia and Switzerland. We reviewed the schemes underlying these systems in section 5 and concluded that they rely heavily on trusted authorities.

B Opinionated thoughts on trust, verifiability and understandability

As we noted in introduction, *trust* has two meanings [73]. Throughout this thesis we have discussed trust from the computer science perspective: as a negative property which should be minimized. Now we will shift to the social science perspective and discuss trust as a positive property which should be maximized – in particular, how can we get the public to trust a voting system?

Understandability of a voting system is often said to be important to build trust and increase citizen participation in the democratic process. A frequently articulated counterpoint against e-voting systems is that a paper voting system is easy to understand whereas an e-voting system is difficult to understand. As a blanket statement this is not true. We will present two counterarguments to demonstrate. The first argument is more of a pedantic argument demonstrating how it is not true in *all* cases. The second argument is more of a practical one.

Firstly, there are different kinds of voting systems. As we have seen in previous sections, not all paper voting systems are simple but some are actually rather complicated. There exist also e-voting systems which are simpler than comparable paper voting systems. For example, we can take any in-person paper voting system and replace the pen with a computer to end up with a more understandable and more secure system than we started off with.

To elaborate on the above example, in the fully paper-based system the voter expresses their intent with scribbled characters on paper and returns the ballot *before* their intent is interpreted. The ambiguity of written characters causes a significant portion of votes to be disqualified during the tally. If we replace the pen with a computer, the voter now expresses their intent on the computer, and the computer prints out a ballot with their interpreted intent. The voter verifies that their intent is interpreted correctly and casts the ballot. Note that the voter casts the ballot *after* their intent is interpreted. If there is any issue with the computer, it will be discovered before it causes any damage. With this system, a voter doesn't need to understand the inner workings of a computer any more than they need to understand the inner workings of a pen. The reason we claim this system to be more understandable than its fully-paper counterpart is that it removes the need for voters to understand which handwritten symbols are acceptable and which are not.

Secondly, voters typically understand paper voting systems at a rather superficial level: Voters understand how the voting booth and the ballot box provide ballot secrecy. Voters understand how election officials prevent people from voting multiple times. But if you ask voters which statistical methods are used to verify the correctness of the tally, or which methods are used to prevent vote-buying, it becomes very clear that essential building blocks of a paper voting system are not common everyday knowledge.

It is our opinion that trust in voting systems *does not* primarily arise from understandability. It is definitely a contributing factor, but primarily trust arises from transparency and the subsequent belief that independent election monitors are able to verify the correctness of the tally. It is our opinion that an e-voting system could achieve this same level of public trust, as long as it provides the verifiable security properties described in section 3. To put it in other words: your grandmum doesn't have to become an expert cryptographer in order to trust a system like X. She just has to believe that cryptography experts exist and at least one of them would speak out if this transparent voting system was not as secure as the election officials claim.

Furthermore, even though verifiability may contribute towards transparency and trust, we do not see verifiability as a means to an end: verifiability should be a goal in and of itself. In fact, verifiability is extremely valuable in a voting system *even if* it decreases the public's trust due to its complexity.

As an extreme example, consider the case where a voting system lacks verifiability, is trusted by the public, and is compromised by a foreign superpower: the people have lost their democracy and do not even realize it. Compare that to a hypothetical case where a voting system has perfect verifiability, thus can not be compromised (without triggering a new election etc.), and, for whatever reason, is *not* trusted by the people.

Clearly, the outcome where people are suspicious of a perfectly functioning voting system is superior to the outcome where people are blindly trusting a compromised voting system. We hope that this outlandish example is enough to support our argument that *verifiability is more important than trust*.

This is a contentious topic. Pieters [73] argues that trust in voting systems arises – not from understandability, transparency or verifiability – but from familiarity. People tend to trust things which appear familiar to them. The implication of this view is that trust in a voting system can be facilitated by designing familiar interfaces and familiar service paths. In contrast, things like cryptographic verification protocols are inherently unfamiliar to normal people.

C Blockchain and P2P voting schemes

In recent years, P2P (”peer-to-peer”) and blockchain-based solutions (some of which are P2P) have been hyped as the solution to remote e-voting. As we noted in section 4, the use case that we see for blockchains within voting schemes is as an implementation for the public bulletin board, but there are simpler solutions for that already and it is unclear what advantages the (much more complex) blockchain solution is supposed to provide. In any case, we decided to perform cursory reviews on several blockchain and P2P voting schemes.

BitBallot

Riemann [78] describes a P2P voting protocol called BitBallot. According to Riemann, the protocol assumes *”honest authority and honest peers”*. This is clearly unacceptable for a voting protocol, which is why we did not investigate BitBallot further.

ADVOKAT

ADVOKAT is another P2P protocol described by Riemann in [78]. It is based on Kademlia (the Distributed Hash Table underlying BitTorrent). According to Riemann, ADVOKAT does not provide receipt-freeness or coercion-resistance, and ADVOKAT assumes that the majority of peers are honest. We decided to not investigate the scheme further because we are unaware of its intended use case. Since the scheme lacks receipt-freeness and coercion-resistance, it would be suitable for low-coercive environments only. However, the main advantage of the scheme is postulated to be its high availability. We are unaware of a use case which is simultaneously low-coercive and requires high availability (higher than what Helios currently provides). Compared to Helios, ADVOKAT has weaker correctness, verifiability, fairness, and confidentiality properties. This tradeoff does not seem worth it for higher availability in a low-coercive environment. Due to these reasons we decided not to look into ADVOKAT in more detail.

BallotCoin (blockchain voting)

We were unable to locate a white paper or website for this project. It appears to have been abandoned long ago. Interested readers will find some descriptions of BallotCoin in [62].

BitCongress (blockchain voting)

The website domain for BitCongress appears to have been repurposed to show blockchain related advertisements. The project appears to be abandoned. We were unable to find documentation for their voting scheme or the source code for its implementation.

BitVote (blockchain voting)

The website for BitVote is still accessible, but the latest update appears to be from 2012. We were unable to find documentation for their voting scheme or the source code for its implementation. However, the donation button on their website still works.

VoteCoin (blockchain voting)

The website and whitepaper are still online, but the project appears to have been abandoned. The whitepaper concerns mainly their Initial Coin Offering. The only advantage that they postulate their voting system to have over traditional voting systems is that the user can divide their vote between multiple candidates. We were unable to find documentation for their voting scheme or the source code for its implementation.

Follow My Vote (blockchain voting)

Follow My Vote[79] is purported as a verifiable voting scheme which uses blockchain as a public bulletin board. On some parts of the website they claim to have open-sourced all of their code. On other parts of the website they say they will open-source the code in the future. We visited their GitHub repository, and at the time it contained forks of unrelated repositories. We were unable to find documentation for their voting scheme or the source code for its implementation.

Agora (blockchain voting)

Agora[80] is another closed-source[81] blockchain project which claims to be working on a verifiable voting system which uses blockchain as a public bulletin board. The design they present in their whitepaper is nearly

[79] https://followmyvote.com/ (accessed 30.7.2019)

[80] We reviewed version 0.2 of Agora's whitepaper at https://www.agora.vote/ (accessed 30.7.2019)

[81] Although a GitHub repository for ”Agora voting” exists, it belongs to an entirely different, unrelated project.

incomprehensible. It involves several different blockchains, networks, applications, and various cryptographic building blocks from academic voting literature. It seems that they wanted to make a soup with every possible ingredient. However, they do not motivate the reader with what they hope to achieve with this design as compared to established systems like Helios or Civitas. The only comparisons they provide are in relation to traditional DRE systems; they simply rehash well known weaknesses of black box electronic voting. We did not see enough merit to warrant a closer look at their complex scheme.

SecureVote (blockchain voting)

SecureVote[82] claims to be another blockchain project for secure voting. We were unable to find documentation for their voting scheme or the source code for its implementation (they focus more on marketing their Initial Coin Offering). They make some wildly untrue claims about other voting systems, which indicate that they should not be taken seriously:

- They claim that manipulating votes is easy in any centralized voting system, even though we have shown several centralized voting systems where manipulating votes is not easy. *"...no one is able to change the vote. This is in stark contrast to any centralised voting system, online or offline, whereby system administrators and officials can easily manipulate or remove votes from the system."*
- They claim that anonymizing votes is an unsolved problem, even though a multitude of voting schemes have been published in academia, which successfully anonymize votes with a variety of methods, including mix networks, homomorphic encryption, and blind signatures. *"To date, all commercial attempts to replicate ballot box voter anonymity in electronic systems have required a trusted entity to perform the anonymisation, or are expensive, slow, and unable to suitably scale. Our patent-pending vote anonymising algorithm, 'Copperfield', makes digital secret ballot possible with a very low overhead."*

Voatz (blockchain voting)

Voatz[83] is another blockchain-related voting project. They claim to have been involved in a real world mobile voting experiment in West

[82] https://secure.vote (accessed 30.7.2019)
[83] https://voatz.com (accessed 30.7.2019)

Virginia with real votes cast through their mobile application. However, their source code and voting protocol are not available for review.

Voatz is a great example of how commercial vendors can fill the letter of the law without filling the spirit: what they refer to as a ”paper audit trail” is literally a physical print of their digital records. If the digital records are corrupted before printing, the paper records will be likewise corrupted.

Based on the descriptions on the FAQ page, Voatz servers are all running the same software, and the voter needs their vote to be recorded on all of them in order to receive confirmation that their vote has been recorded. This seems like the worst of both worlds: a single misbehaving authority can prevent the election from proceeding, and at the same time, the ”verifiable blockchain” benefits are undermined by running the same closed-source unverifiable software on all of the servers and the same closed-source unverifiable software on all of the clients. The replication of servers provides security benefits only against physical tampering of the machines – since the same software is running on all of them, an accidental or intentional flaw in the software can be exploited to manipulate votes on all servers simultaneously.

Polys (blockchain voting)

Polys is an interesting case. It is backed by reputable Russian company Kaspersky. However, their website's ”whitepaper”[84] indicates that it is a traditional ”black box” commercial voting system with a little bit of blockchain sprinkled on top. To elaborate:

- Their voting protocol is not documented in the whitepaper.
- According to the whitepaper, their source code is not auditable: *”We are preparing to make the Polys protocol code open source and plan to publish it on GitHub in the near future. We are currently auditing the Polys protocol code internally.”.*
- While they are actively selling their ”black box” software to clients, they bemoan the dangers of (other providers') black box voting software: *”...voting cannot be carried out in black-box mode — the process should be clear and transparent for all participants”.*
- They appear to be blissfully unaware of the progress made in receipt-free and coercion-resistant voting systems: *”How do you*

[84] The Polys ”whitepaper” is simply a FAQ section on their website (not a PDF), so it will likely evolve over time. We viewed the website as it appeared on 18/9/2019. https://docs.polys.me/en/collections/699457-technology-whitepaper

protect people from external pressure or the temptation to sell their votes? Importantly, we still want to have some kind of mechanism to verify that a vote has been counted — and if such a mechanism is in place, it can be used for trading in votes."

- They currently provide no information how a stakeholder could verify that the election was conducted honestly. Even if the closed-source platform provided by Polys is operating honestly, the authorities running the election still have the ability to manipulate the election result (by creating voter credentials for non-existing voters) or break ballot secrecy (by decrypting individual votes after creating tallier credentials for non-existing talliers). They claim to use a threshold decryption system. Even if honest talliers are present, Polys' documentation does not indicate if honest talliers have a method of verifying whether tallying credentials have been generated for non-existing talliers or not.

D Reviews of similar prior work

In this appendix we take a closer look at similar prior work from the perspective of our research questions. We focus on two things: comparisons and their underlying comparison frameworks. In some cases we highlight differences between this thesis and similar works. In many cases the authors' scope is different from ours and they do not provide an actual comparison, even though they provide (what could be construed as) a comparison framework. This appendix exists to justify why a comparison of voting schemes' security (this thesis) was needed.

A framework and taxonomy for comparison of electronic voting schemes

In [85], Sampigethaya and Poovendran present a great taxonomy of voting schemes and their security properties. Their comparison table shares many similarities with Mursi et al.'s [64] comparison table. Sampigethaya's and Poovendran's work is very thorough and they have compared a wide variety of different schemes. However, the article was written in 2004, so their comparison does not include modern schemes, which we compare in this article. Furthermore, their comparison is limited to remote e-voting schemes, whereas we compare other kinds of voting schemes as well.

We consider Sampigethaya's and Poovendran's comparison framework to be the pinnacle of comparison frameworks for voting schemes. However, it is not without faults and we hope to surpass it with our framework, by building on top of their ideas and the ideas of everyone else in the 15 years since Sampigethaya's and Poovendran's comparison framework was written.

The properties under comparison are carved out differently in our framework compared to theirs – they are comparing familiar concepts from voting literature, whereas we attempt to carve out properties to maximize a specific set of goals (section 3 describes our goals and how the properties in our comparison relate to familiar terminology in voting literature). In addition, we are more explicit with our assumptions and the values in our comparison table are less ambiguous (for example, many cells in their tables have value ”conditionally satisfied”).

A taxonomy and comparison of remote voting schemes

In [57], Li et al. compare a dozen voting schemes' most important security properties. Unfortunately, this is not an apples-to-apples comparison because each scheme is reviewed under a different set of assumptions (which are not documented in the article, but are presumably discoverable via references).

Furthermore, the comparison has many inconsistencies and the claims presented by authors are weakly justified. For example, Li et al. define universal verifiability as simply *”Anyone can verify the final voting results”*. One of the reviewed schemes, Cobra, is described with ”The final tally is verifiable”. However, in the comparison table the authors have marked ”No” under ”Universal verifiability” for Cobra. Based purely on the definition presented and the description of Cobra presented we would have expected this to be a ”Yes” instead of ”No”. Perhaps this is an error, or perhaps the tally is verifiable by privileged auditors, or perhaps there is some other explanation. Making sense of these conclusions is very difficult when the authors have not properly justified them. We noticed a similar issue with Civitas' Universal verifiability. Although they describe Civitas with ”Tabulation is publicly verifiable”, the value presented in the comparison table is ”Not Known”. Again, no justification is provided.

Design of distributed voting systems

In [62], Meter describes several important security properties, but the descriptions are rife with errors.[85,86] Meter claims to present a novel voting scheme with superior security properties compared to other schemes (comparison table on page 87). However, upon closer inspection we realized Meter *actually doesn't present a voting scheme.*[87] It is unclear how Meter produced values for his comparison table.[88]

Meter provides an additional comparison table on page 44, this time with many confusing properties instead of the security properties discussed earlier. As an example, the values for *"Encryption"* property are either "Public Key" or "ElGamal". But the El Gamal cryptosystem is also a public key cryptosystem, leading us to wonder what the author wishes to express with this comparison.

Desirable properties of voting systems

In [60], Lowry and Vora offer a good overview into different kinds of voting systems, with examples of the most famous voting systems of each kind. Their scope is larger than ours (for example, they cover usability and accessibility) but they only scratch the surface of most topics. We have a much narrower scope, but we go in depth to the topics that we cover.

Lowry and Vora provide good justification for what they see as the most desirable properties in voting systems. However, they do not treat their collection of desired properties as a comparison framework, and they provide very few, limited comparisons.

[85] For example, Meter erroneously equates *coercion freeness* with *receipt-freeness* despite that he lists [45] and [20] in his references (both of these papers clearly distinguish coercion-resistance and receipt-freeness).

[86] As another example, Meter erroneously describes *Voter Verifiability* in the following manner: *"The voter herself must be able to verify that her ballot arrived in the ballot box. This ensures that the voter is sure her vote was counted and was not modified."* However, as we have demonstrated numerous times, this recorded-as-cast verification does *not* ensure that the vote will be counted.

[87] Section 5, which purportedly contains Meter's scheme, is actually a discussion on various high-level ideas, such as the merits of web applications versus native applications, or the merits of open source versus closed source. Strangely, even though Meter claims to present a novel voting scheme, he actually does not present a voting scheme at all.

[88] In the case of Meter's purported voting scheme, Meter provides some reasoning for the values, but the claims can not be verified because the scheme is not described anywhere. In the case of *others'* voting schemes, the values presented in the comparison table are not explicitly justified, but can sometimes be inferred from text in various locations of the article.

Dutch e-voting opportunities. Risk assessment framework based on attacker resources.

In [100], Verbij provides a good overview into security properties described in literature. On page 52, Verbij presents a comparison table. The comparison covers many relevant security properties, although many important properties are missing.[89] The properties also have some unnecessary overlap.[90] The comparison covers only 3 voting schemes (Estonia, France and Norway).

Additionally, Verbij proposes a framework for the purpose of identifying ”realistic risks in e-voting schemes” (not directly related to the security properties discussed earlier). However, it would appear that their framework is limited to cost estimations of traditional malware attacks (rather than the general ”realistic risks” scope declared by the authors).[91]

On the Development of Electronic Voting: A Survey

In [64], Mursi et al. provide both a comparison framework and a comparison. The comparison table and related taxonomy of voting schemes share many similarities with [85]. They present a list of security requirements discovered in literature, but they do not attempt to consolidate all of the overlapping and contradicting requirements.[92] Furthermore, they claim that their list contains formal definitions, but it actually contains informal definitions and many of them are vague and ambiguous.[93]

There is a comparison table in section 6. However, the authors do not compare actual schemes to each other – instead they compare what they call ”classes” of schemes. For example, one class is called ”Hidden vote, authority key threshold”. The authors do not explain how they derived the values in the table. In addition, the authors claim to

[89] For example, Verbij doesn't compare eligibility verification, availability and dispute resolution properties.

[90] For example, Verbij presents cast-as-intended and contains-correct-vote as distinct properties.

[91] This is our impression based on the authors' examples of applying the framework. We do not understand their framework and the article is 195 pages long. We may be wrong, but in any case, their framework is fundamentally different from ours.

[92] For example, Mursi et al. require that participation rolls should be public (”Verifiable participation”) and, at the same time, participation rolls should be secret (”Incoercibility”). Both of these outcomes can not be desirable simultaneously. To be fair, Mursi et al. acknowledge that the list contains contradictory requirements.

[93] As an example of an ambiguous definition, Mursi et al. define *Verifiability* as *”Voters shall be able to verify that their votes are correctly counted for in the final tally (universal or individual)”*. However, they do not define what they mean by these terms.

present a ”Comparative analysis of schemes” in section 8, but section 8 does not actually contain a comparative analysis of schemes (it contains descriptions of schemes without making comparisons).

Public Evidence From Secret Ballots

In [8], Bernhard et al. call for evidence-based elections. The authors describe important security properties, provide case reviews of voting systems, explain common building blocks in voting schemes, and summarize their findings in a polished comparison table. In other words, their structure and scope is very similar to ours. However, the article is rather short, so they only scratch the surface of most topics.

The results presented in their comparison table appear to contain several errors. For example, they erroneously claim that Civitas provides cast-as-intended verifiability, even though it is disabled by design, as we explained in section 5.9. As another example, they erroneously claim that ThreeBallot provides count-as-recorded verifiability, even though it does not[94]. These claims are not justified in the article in any way. Even the references the authors provide for ThreeBallot and Civitas do not support these claims. We reached out to one of the authors and did not hear back.

Furthermore, some of the properties in their comparison are ambiguous. For example, one property is called ”take-home evidence”. That could mean many things depending on the author and the term appears only once in the entire article – there is no description or references. The authors also excluded some crucial properties from the comparison with dubious justification: *”certain features [like] availability are excluded, as these factors impact all systems in roughly equivalent ways”*. As we clearly demonstrated[95], this claim is false. Availability does not impact all systems in roughly equivalent ways and it is a crucial feature which must be considered in any holistic comparison.

[94] As we noted in section 5.2., ThreeBallot is vulnerable to a wide range of collusive attacks.

[95] For example, we showed that Civitas has severe availability issues, which do not affect the Finnish paper voting scheme. We refer to sections 3, 5 and 6 for details.

Source code review of the Hart InterCivic voting system

In [41], Inguva et al. provide an excellent threat model and detailed classifications for attack types, attacker types, and many related subjects. They provide a great framework for analyzing a voting system, although their framework is tailored towards one specific voting system which they analyze, and their analysis involves many practical aspects which are outside of our scope.

Survey on electronic voting schemes

In [29], Fouard et al. present a great overview into different security properties defined in literature. They even documented how the same terms are defined differently in different works. Unfortunately, they overlook some crucial properties: dispute resolution is not addressed in any way and coercion-resistance is presented as if it were equivalent to robustness, for some reason. After listing all these different definitions, the authors consolidate them into a coherent comparison framework (although they do not justify their decisions in this step).

There is a comparison table on page 61. Unfortunately, the authors do not present schemes under unified assumptions – instead, each scheme is evaluated under a different set of assumptions (omitted from the article, but presumably articulated in the referenced articles). As such, this is not an apples-for-apples comparison. Additionally, the table has values which are difficult to interpret without substantial research.[96] Furthermore, as the article was published in 2007, the selection of schemes for comparison is markedly different from ours.

[96] For example, many cells in Fouard et al.'s comparison table have value "A", as in "attacks found". How significant are these attacks? Do they entirely break the property in question or are they minor infractions in special circumstances? Impossible to say without substantial research.

Towards Trustworthy Online Voting: Distributed Aggregation of Confidential Data

In [78], Riemann covers most of the important voting security concepts in literature, but does not consolidate them into a coherent framework. Properties are presented as disparate concepts even when they have obvious overlap[97]. Furthermore, some of the definitions presented by Riemann are ambiguous and we disagree with some of them.[98]

Although Riemann does cover important voting security concepts, he does not use these concepts to compare the security of different voting schemes. This point is not a criticism of his work; his scope and focus is simply different than ours.

[97] For example, Riemann defines ”individual verifiability” in a short description. Later, Riemann defines ”End-to-end verifiability” as consisting of ”cast as intended”, ”recorded as cast” and ”tallied as recorded”. Then Riemann defines ”software independence” in abstract terms. As we demonstrated in section 3, these terms have significant overlap and it is unhelpful to the reader to present them as disparate concepts.

[98] For example, end-to-end verifiability is usually defined as the combination of individual verifiability and universal verifiability, but Riemann does not include universal verifiability in his definition for end-to-end verifiability. No justification is given.