ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Swiss Internet Voting

Master Thesis

Florian Moser

February 16th, 2022

Advisor: Prof. Dr. Kenny Paterson

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

In the country of Switzerland, there have been numerous attempts to introduce internet voting in the past two decades. Current approaches rely on purpose-specific complex cryptography to resolve the seemingly contradictory security requirements of verifiability and privacy. This results in systems which are very expensive to understand, proof, develop and review.

We performed a survey of the literature and of history, law and politics in Switzerland concerning vote électronique. Based on this background, we reach conclusions how the situation can be improved to lead to securer systems. This culminates into a new proposal for a vote électronique system using code voting, drastically reducing the complexity of the involved cryptography while reaching stronger security properties. We present formal definitions for the requirements set forward by law, and prove our construction to fulfil these formal definitions using a computational proof.

Contents

Chapter 1

Introduction

In Switzerland, there exist two ordinal voting channels: Postal voting and in-person voting at the polling place. Around 90% of voters use the postal channel [99]. Additionally, Switzerland is attempting to introduce an internet voting channel, primarily due to its advantages regarding usability and availability [37, 47]. Serious efforts started as early as 2001 [54], and a clear solution has not yet established.

This work takes a holistic approach to internet voting for Switzerland. We first survey relevant international literature and then summarize the history, politics and legal situation of internet voting in Switzerland. Based on this understanding, we propose a new internet voting system: While simpler than existing proposals, it achieves even stronger security guarantees. Its core mechanism, previously not considered for Switzerland, is code voting: Instead of entering their choice directly, like ”Yes” or ”No”, voters enter codes corresponding to their choice, like ”2” for ”Yes” or ”3” for ”No”.

Whether this results in an overall favourable system is then again - as well as whether an internet voting channel should be provided at all - a political decision which we do not aim to to answer in this work.

Structure of this work To let the reader easier navigate this more work more easily, and find what is important to them, we provide a short overview of each chapter. All chapters approach electronic voting from their own perspective, and could be considered independent contributions. Together, they form one larger, informal argument: The proposal we define and prove as part of this work is optimal within the constraints imposed by literature and reality in Switzerland.

In chapter 3, we survey literature. We first focus our attention on the privacy and verifiability properties commonly achieved by electronic voting protocols, how these properties are understood differently by different authors

and how the properties might interact with each other. Then, we survey notable electronic voting schemes, with a particular focus on remote schemes proposed by research as well as industry. We then classify and describe mechanism which are in use to reach the privacy and verifiability security properties.

In chapter 4, we observe how electronic voting impacts Switzerland. We reconstruct the history of electronic voting. We then analyse the current legislation and set it in context with international law. We collect arguments in favour and against vote électronique from the current political debate in Switzerland. Lastly, we summarize the three conceptually different approaches of multiple cantonal administrations to provide an electronic voting system to their respective citizens.

In chapter 5, we summarize the setting of an internet voting system in Switzerland. We first define the restrictions imposed by law and reality in Switzerland, concerning involved parties and legally required security properties. We then argue what minimal trust assumptions cannot be eliminated, what maximal security properties we might aim for, and how one could reduce the overall complexity of such a system.

In chapter 6, based on the insights of the previous chapter, we iteratively construct a protocol we argue to be optimal under the aforementioned constraints. Focusing on the voting phase, we establish a verifiable and a privacy-preserving proposal. Then, we join these two proposals together and define an appropriate setup and tally phase.

In chapter 7, we formally describe our proposal at a level of detail sufficient for an implementation. We describe the infrastructure we rely on, which simplifies the description of our protocol. Then we describe the protocol, split into setup, voting and tally phases. For each non-trivial operation, we provide pseudo-code. Finally, we argue how the protocol can further be extended.

In chapter 8, we describe a formal model for electronic voting protocols and security properties. We first define the roles, terminology and syntax of this formal model. Then we provide game-based definitions of verifiability and privacy guarantees, which match the requirements put forward by Swiss law.

In chapter 9, we prove our proposal secure. We describe how the infrastructure we rely upon impacts our proofs. Then we instantiate the formal model we defined in the previous chapter, implementing the syntax the formal model requires. Further, we instantiate the security properties and provide tight bounds on the attacker's advantage.

In chapter 10, we reflect on the work as a whole, discuss future work.

Contributions The thesis is comprised of several contributions:

- A literature survey about properties, notable schemes and mechanisms in electronic voting.
- An analysis about how electronic voting impacts Switzerland.
- A discussion on whether and how the current setting of internet voting in Switzerland can further be improved.
- An iterative construction of the resulting protocol, which clearly shows why certain messages are added to the protocol.
- A precise proposal using code voting which turns out to be much simpler than previous proposals for Switzerland.
- A formal model which precisely specifies the requirements detailed by Swiss law.
- A computational proof of the proposal in that formal model.

The thesis as a whole is further one large argument that the proposal is optimal under the imposed constraints.

Chapter 2

Motivation

The history of internet voting in Switzerland is a turbulent one. Since the first trials run in 2003, Switzerland has seen many legislative changes and no less than five different internet voting systems. At its height, half of the cantons had legally binding votes for a fraction of their citizens. The systems and the applicable legislation evolved multiple times, continuously requiring more and more rigid security properties. In 2019, the last two remaining systems found an end: The parliament of Genève denied a financing required to continue development of its system, and the system of Swiss Post was taken offline after it failed a public scrutiny test. Thus, after almost 20 years of trials, Switzerland has now much stronger legislation, but no available system able to fulfil it.

Research has to build around unsolved problems to fulfil the seemingly contradictory requirements (like ensuring the server correctly stores one's own vote, without jeopardizing its secrecy), and resorts to propose largely unrealistic schemes with lacking usability, excessive performance demands, overly high complexity or a combination of all three. Politics has failed so far to set a stable legislative basis and provide sufficient support for internet voting projects to emerge and flourish. Meanwhile, industry and administration struggle to implement such a complex system under continuously changing requirements while choosing the right tradeoff between quality, transparency and economic considerations.

In time, the issues in research, politics and industry might be resolved. In the meantime, however, internet voting will stay in demand, which might provoke only partially satisfying solutions to be deployed. This master's thesis therefore searches for an alternative: Given the internet voting literature, and the experiences with internet voting in Switzerland, what tradeoffs exist and how can they be chosen to lead to a both simpler and more secure voting system?

Part I

Background

Chapter 3

Literature

In its most general understanding, *E-Voting* includes all forms of voting where some electronic device is involved, notably voting facilitation devices or devices to speed up tallying [209]. Research around e-voting explores how one can vote using electronic aids, while providing *Privacy* (ensuring the vote stays secret) and *Verifiability* (ensuring the vote is processed as expected) under realistic assumptions. Additionally, general literature to secure systems applies; specifically concerning *Availability*, *Usability* and *Transparency*.

We focus here on the system the voter uses to cast their vote, starting with the setup of said system, and ending where the tallied votes are output. For the schemes used by these voter casting systems, we differentiate between supervised and remote voting schemes. For *Supervised Voting*, it is assumed the voter can be forced to follow a specific process, like at the local polling station of their municipality. For *Remote Voting*, the voter may (ab)use the provided system in any way they can, while not being constrained to specific polling places. We focus on the latter, even more specifically *Internet Voting*, where the vote is cast over the internet.

History of e-voting research In 1981, Chaum introduces mixes, and mentions e-voting as one of the applications [73]. In 1985, the first e-voting scheme using encrypted ballots is introduced [77], then a second category using anonymous channel follow in 1988 [67]. In 1992, the first scheme based on blind signatures is published [124]. In 1994 and 1995, two fundamental e-voting specific properties are introduced for privacy and verifiability, respectively: Receipt-freeness and universal verifiability [22, 259].

First, the community focused on hardening the privacy of voting schemes [220]. An example is the FOO scheme based on blind signatures already proposed 1992 [124]. The property of receipt-freeness, proposed in 1994 [22], motivated additional schemes [220]. Privacy notions further strengthened

with the proposal of coercion-freeness in 2005 [157] and everlasting privacy in 2006 [194]. At the same time, the focus of the community began shifting towards strengthening the verifiability of the systems [220].

Motivated by Chaum proposing visual cryptography (hence executable by the average voter) in 2004 [69], these schemes aimed at being practical, besides strong verifiability guarantees [220]. An example of a verifiable scheme with a very simple concept is Prêt-à-voter from 2005 [72] which would then be gradually improved over nearly a decade. In this period also the internet voting scheme Helios is proposed, including an implementation easily accessible to the public [2].

Finally, the first systems with strong verifiability guarantees developed by researchers were starting to be used in real-world governmental elections. Examples for electronic voting machines include Scantegrity, first used in 2009 [70], and vVote, used in 2014 [62, 90], with both based on the Prêt-à-voter idea [220]. Internet voting schemes were tested on student associations, such as in Switzerland [140, 103, 102] and Austria [165], but then did not find their way into governmental elections.

In 2011, the first protocol developed by industry with some notion of verifiability was published and then used in elections in Norway [7]. Although criticised as of low quality [217], similar systems (facing similar critique) by the same vendor Scytl were then introduced in Switzerland and Australia (see section 3.2.3), besides other countries.[1] Estonia is the only country which has been able to provide internet voting for more than a decade, having continuously adapted its system to the new developments in research [282]. In Switzerland, multiple different systems were used over time, with often changing functionality and applicable legal base (see chapter 4).

In 2016 and 2017, an effort was made to clearly list and define properties required of an e-voting system [84, 26, 168]. Further, over the last years cryptographic and symbolic proofs have become more common, both for existing systems used in industry (e.g. [88, 262]) as well as directly published together with new schemes [169, 14, 141].

3.1 Properties

We start by reviewing properties an e-voting system should provide.

We categorize these properties into privacy, verifiability and general security requirements. Privacy is concerned with keeping the voter's intention private, to protect its vote from coercion. Verifiability ensures the voting system works as expected, to bolster trust in the system and prevent or enable

[1] Scytl remains the only international firm providing internet voting to governments.

detection of tampering. The general security requirements are not specific to e-voting but apply to all secure systems, to enable convincingly secure development, review, operation and maintenance.

Verifiability and privacy properties may conflict with each other, making it difficult if not impossible to fulfil all of them at the same time [74, 286]. In general, it can be summarized that absolute privacy (no information leaked) cannot coexist with verifiability (sufficient information available to verify the result) [155].

Which selection of properties are important, and their exact definition, is unfortunately not established yet. We choose as our primary source in this chapter a survey from 2017, with the authors being a who's who in electronic voting [26]. However, these definitions are not very exact, and the list is not complete (notably, fairness is missing). We therefore refer to additional surveys [155, 84, 168, 123, 249], and papers discussing specific properties.

High Stakes and Strong Attacker Simple and feature-rich e-voting systems could be easily designed if some parties were to be assumed trustworthy. However, depending on the magnitude of decisions taken by the system, such strong trust assumptions become unrealistic: For decisions of national or international relevance, nation state attackers must be assumed, with their respective enormous resources and influence. Further, voters might not agree upon which authorities are trustworthy and hence might distrust an election result supervised by the "wrong" authority.

To counter malicious authorities, *Distribution of Trust* is proposed: Multiple authorities perform crucial tasks together, such as to decrypt the election result. The property in question is then only broken if all or a significant portion of the authorities work together on the subversion. Choosing sufficiently many, sufficiently heterogeneous authorities ensures collaboration becomes extremely unlikely, and therefore the property remains preserved.

Besides untrustworthy parties, the used tools could be malicious or simply faulty. The *Secure Platform Problem* recognises this unobservable black-box fashion in which hardware and software operate [244]. This holds for the devices employed by the authorities, but also for the personal devices used by the voter. Indeed, even a cautious voter cannot reasonably defend its personal device against a targeted attack by a properly motivated and funded attacker.[2] The voting authority devices pose an even more valuable attack surface, while not having access to much stronger defence mechanisms.

To counter bad tools and devices, a concept called *Software Independence* was proposed. It requires that an (undetected) change or error in software cannot cause an undetectable change or error in an election outcome. Weak

[2]For a recent example, see the Pegasus leaks https://www.nzz.ch/ld.1640310 (2021).

software independence is achieved when misbehaviour is detected, while strong software independence allows to correct the detected error without having to rerun the election [246, 248]. Necessarily, all data required to calculate the end result must be published.

3.1.1 Privacy

Privacy properties ensure the vote stays private to the voter. This enables free expression of opinion, without fear of retaliation before, during or after the election. Even voters intending to prove their vote to an attacker (for example, to win some cash prize) are not convincingly able to do so when the strongest privacy properties are fulfilled.

There are however inherent limits to how far privacy can go; the stronger the privacy notion, the more complex (or even impossible) it becomes to reach verifiability properties. Acceptable privacy notions may also differ depending on the electorate and the stakes of the election.

Ballot Privacy, hence that the vote is not revealed to anyone, seems to be the absolute minimum privacy property to aim for, already recognised by the first schemes in 1985 [77].[3] The first schemes claiming to implement the property opted for anonymous channels [67] or encryption of the vote [77].

Over time, the privacy properties got stronger and stronger. *Receipt Freeness* was introduced in 1994, and guarantees the voter cannot prove how they voted (even if they wanted to) [259]. In 2005, Jules et al. propose *Coercion Resistance*, which essentially requires the voter actions to be indistinguishable to the attacker [157].

Benaloh argues in 2013 that the strong privacy notions literature aims for at that time might be too strong. When using the (physical or virtual) voting booth, coercing voters could simply create video proof of their vote, defying any mighty privacy mechanism implemented within. He argues the attack to be scalable, as the attacker can probabilistically check the video proof.[4] Benaloh observes pre-registration coercion as practically impossible to defend against, and declares it a "lost element". Instead, protocols should focus on preventing post-election coercion, and avoid unnecessary complexity archiving close to no benefit [20].

When looking at the schemes employed by industry, receipt freeness after finishing the voting phase is indeed provided, although by sacrificing verifiability properties. Concretely, only the election result is published, but

[3] Note that the formulation "not revealed to anyone" is clearly too strong, as the vote has to be at least revealed to the tallying authority in some form.

[4] When randomly auditing 5% of all supposedly coercing voters, then watching their video proof at 5 times the recording speed, the attacker is 100 times faster at verifying than the voters were at proving.

no individual recorded votes. Auditors ensure that the individual recorded votes actually sum up to the election result, but the general public is not able to.[5]

Somewhat separate - and uncontested - is *Fairness*, which requires no one learns (parts of) the election result until the count. It essentially enforces that votes must be encrypted if they are transferred over an insecure network or to an untrusted authority during the voting phase.

Also somewhat separate, but much more difficult to achieve, is *Everlasting Privacy*, which requires the privacy guarantees to not erode over time. Concretely, the encryption (or whatever binds the vote to a voter) being used must not be based on a cryptographic hardness assumption.

Ballot Privacy

> A voter's vote is not revealed to anyone [267].
>
> No outside observer can determine for whom a voter voted [155].

This most basic privacy definition was how privacy started in e-voting [155]. Even the earliest schemes from the 1980s achieve this property, and see it necessary to do so [77, 67]. It can be directly translated from article 25 c) of the UN human rights pact II [97]. While also accredited by Switzerland, a secret vote is not required by its constitution [105], and it depends on the canton whether this is implemented.[6]

The property provides privacy in intimidation-free environments and is considered the *de-facto* standard privacy requirement [267]. Most recent schemes aim for stronger privacy properties, to support voting even in environments with coercion.

Exactly defining ballot privacy can be tricky; definitions might be too weak or strong, or simply not apply to a wide area of protocols [23]. Formal privacy proofs were for example done for sElect [169] or Helios [23].

As demonstrated by the first schemes claiming to implement this property, anonymous channels are enough to support this property [67], even under strong verifiability guarantees. Alternatively, of course encrypting the vote also ensures its contents stay private [77].[7]

Receipt Freeness

> A voter cannot prove after the election how it voted [155].

[5] Like the Swiss Post protocol from 2021 (see section 3.2.3).
[6] Some cantons have *Landsgemeinde* where the electorate votes publicly by show of hands.
[7] Assuming the decryption keys are properly managed.

> A voting system is receipt free if a voter is unable to prove how she voted even if she actively colludes with a coercer and deviates from the protocol in order to try to produce a proof [26].

Receipt freeness was proposed in 1994 to counter what is now known in the literature as *Italian attacks* or a *failure of the short ballot assumption*: With many potential voting options available, voters can be coerced to vote a specific combination. If this combination does not appear in the final tally, then the voter did not comply [22, 247, 26].

It is difficult to achieve, as it often conflicts with verifiability (notably accountability) guarantees, which might require providing a receipt to the voter. These receipts must therefore be designed in such a way that they do not reveal the voter's intention, which introduces complexity and hard-to-spot edge-cases.[8]

Coercion resistance

> A voter cannot interact with a coercer during the election to prove how she is voting [155].
>
> A voting scheme is coercion resistant if there exists a way for a coerced voter to cast her vote such that her coercer can not distinguish whether or not she followed the coercer's instructions [26].

Coercion resistance was introduced in 2005 to account for more powerful attacks previously not considered: Besides forcing a specific vote, the attacker might also force abstention or randomization of the vote, or even surrender of credentials [157]. The voter should be able to submit its true preference, even when being monitored by the attacker [75, 166]. Or differently formulated, the coercer should not be able to distinguish whether the voter followed the instructions of the adversary or not [277].

Various approaches exist to fulfil this property, such as revoting, or faking credentials, votes or receipts [166]. However, the complexity introduced is usually considerable, and it is very difficult to provide usability and verifiability nonetheless. Consider the JCJ/Civitas scheme, which is usually used as a prime example of a scheme archiving this property: In a usability study, 90% of the voters failed to understand how the fake votes worked [198]. Further, concerning verifiability, it can be argued that cast-as-intended is not provided as the voter does not (and indeed must not) get feedback whether it cast a fake or valid vote [158].

Fairness

[8] For example, ThreeBallot receipts still allow to trace the vote in large electorates [245].

> No participant can gain any knowledge, except his vote, about the (partial) tally before the counting stage [249].

This property is recognised to be necessary for even longer than receipt freeness. Already in the paper for the FOO scheme of 1992, the authors consider fairness an important property and lament previous work that does not achieve it [124]. The property ensures voters voting late have no more information available about what is in the tally than those voting early.

The property is not particularly hard to achieve, for example using a threshold decryption schemes on the encrypted votes in the end. However, it might still prevent some particularly inefficient schemes to reach production: Counting procedures of multiple days might be unacceptable to the electorate, while fairness prevents the counting to start before the election ends [272].[9]

Everlasting Privacy

> A voting scheme has everlasting privacy if its privacy does not depend on assumptions of cryptographic hardness [26].

The idea was introduced in 2006 with a corresponding fulfilling voting scheme [194]. The idea is to keep a vote secret indefinitely, avoiding any consequences for the voter even in the far future. It can be argued current voting reasonably offers this property, as the votes are destroyed after the count. Voting schemes that rely on destruction of data to uphold the property were proposed to be referred to as archiving the weaker *Practical Everlasting Privacy* [9].

A core observation is that a future attacker might has access to much more computational power, but to less data (due to its permanent destruction) [9]. Another observation is that only privacy has to hold unconditionally; verifiability guarantees may erode over time (as long as they do not break until after the election is permanently decided) [91].

This feels like a perfect example of applying a perfectly hiding / computationally binding commitment scheme! However, in these proposals the vote still requires encryption and, if this encryption is based on computational assumptions, only authorities must learn the vote that are trusted to destroy it after the election [195, 96]. Other schemes may rely on an anonymous channel to the voting authority [177, 180], but an implementation of an anonymous channel might again use computational assumptions [92]. Another primitive which unconditionally preserves privacy and might therefore prove itself useful is perfect encryption.

[9] Note that simply detaching the vote from the voter is not enough (for example by using anonymous channels [67]); the votes themselves have to stay intransparent to any untrusted authority until the end of the voting phase.

3.1.2 Verifiability

Verifiability properties ensure the votes are processed as expected. There should be no doubt that the election result is correct, with placing as little trust as possible in authorities, implementations or hardware. The full electorate should be convinced of this fact, both winners and losers, so everyone accepts the election outcome.

Verifiability notions were first introduced in 1995, as *Individual Verifiability* and *Universal Verifiability* [259]. As terminology implies, it was somewhat focused on *who* performs which checks. The voter verifies that their vote reached the destination, and everyone else verifies that all votes are summed up correctly.

After almost a decade, the notion of *End-To-End Verifiability* started to establish [155]. It seems to be first mentioned in 2004 [197], although it is not explicitly explained there. Nowadays, besides different understandings how end-to-end verifiability is supposed to be defined [84, 26], research seems to agree that this is the verifiability notion to be pursued (rather than "only" individual and universal verifiability) [168, 26, 84].[10]

However, there is still something missing: Voter authentication. Only eligible voters should participate, else the election result does not represent anything useful [127]. The formalization is known as *Eligibility Verifiability*, and, together with end-to-end verifiability, recognised as another must-have of verifiable elections [26].[11]

Depending on the voting scheme, additional properties might be applicable than those described here.[12] For example, if the protocol tallies using homomorphic encryption, *Ballot Verifiability* - "all ballots that are confirmed contain correct votes" [24] - is additionally required, so voters cannot for example include negative values for candidates [168].

Individual Verifiability

> A sender can verify whether or not his message has reached its destination, but cannot determine if this is true for the other voters [259].
>
> Each eligible voter can verify that his vote was really counted [249].
>
> A voter can verify that the ballot containing her vote is in the published set of "all" (as claimed by the system) votes [155].

[10] Note that indeed most impactful researchers are included in these references.

[11] Some authors imply that it is part of their definition of end-to-end verifiability [84].

[12] Again, some authors include all properties required for a correct election result already in their end-to-end verifiability definition [84].

The concept was introduced in 1995[13] [259]: Voters should be guaranteed the vote actually reached its intended destination. This can only be done by the voter, as no one else could (or should) know how which vote was cast. Mapped to the physical space, this property is trivially fulfilled when a paper vote is put into the ballot.

Some definitions explicitly state the ”published set” as a bulletin board [162, 268]. These authors are usually concerned primarily with proving, and the way ”published sets” are usually implemented there is by bulletin boards.

In practice, this property is achieved by providing the voter with a receipt depending on the cast vote. The system would ensure that if the receipt is successfully constructed, then some honest authority had to be involved. This mechanism might be combined with another round-trip for the voter to confirm that indeed the receipt was received successfully. Depending on the receipt, these may even be used later by the voter to complain if their vote is not found within the set of accepted votes (which however somehow contradicts receipt freeness).

Universal Verifiability

> In the course of the protocol the participants broadcast information that allows any voter or interested third party to at a later time verify that the election was properly performed [259].
>
> Any participant or passive observer can check that the election is fair: the published final tally is really the sum of the votes [249].
>
> Anyone can verify that the result corresponds with the published set of ”all” votes [155].

Introduced in 1995, it was to ”audit an election, checking it was fair, without getting back in touch with all of the voters” [259]. Essentially, it requires all collected votes are counted correctly, and that this fact can be observed. This is comparable to the physical setting where the ballot is opened and everything inside is counted, under the watch of auditors. The definitions usually require the slightly stronger notion that *anyone* (not just the auditors) can verify the tally.

This property is trivial to implement if all collected votes can be published. Depending on the scheme however, the collected votes are still associated to voter identifications. Then, verifiable shuffles can be used to separate this

[13] Note that some authors claim [155] it was already introduced in 1981 by Chaum [73], although we do not see this connection.

connection, or, if an applicable encryption scheme was used, the votes can be counted homomorphically (and then only the final result is decrypted).[14,15]

End-To-End Verifiability

> A voting system is end-to-end verifiable if it has the following three kinds of verifiability:
>
> - Cast-as-intended: Voters can independently verify that their selections are correctly recorded.
> - Collected-as-cast: Voters can independently verify that the representation of their vote is correctly collected in the tally.
> - Tallied-as-collected: Anyone can verify that every well-formed, collected vote is correctly included in the tally.
>
> [26]

> End-to-end verifiability ensures (...) voters that followed the procedure are guaranteed that their vote is counted in the final result [82].

The first definition, by Bernhard et al., explicitly states individually provable sub-properties, and then defines the system end-to-end verifiable. The idea was around for some time, already Prêt-à-Voter used a very similar structure to motivate its security [72].[16] The definition exists in many slight variations [155, 221], both concerning terminology (for example, collected is sometimes referred to recorded [221]), and meaning (for example, some require tallied-as-collected only to be verified by the voter [155]). The three properties implying end-to-end verifiability is sometimes called the *chain-of-custody* [24].

The second definition stems from the more formal camp. It is more high-level, and explicitly captures the end goal. They argue that the combination of the three properties as given by the first definition is not yet proven to imply end-to-end verifiability, and therefore should not be used [84].

As to the relationship to individual verifiability and universal verifiability: Küsters has proven that they are neither necessary nor sufficient to imply end-to-end verifiability [168]. They are only sufficient with two additional assumptions: When no clashes can be produced (hence the attacker cannot

[14] Note that both of these approaches have performance and/or expressiveness drawbacks. When a voting system is not practically viable, usually this is the step where it shows.

[15] Separating the voter identification from the voter might not be enough to preserve privacy, as discussed already in the context of the short ballot assumption.

[16] With ideas about the sub-properties going back to SureVote [68] and MarkPledge [4] as described in [155].

use the same vote to convince multiple voters of individual verifiability) and when the bulletin board can be parsed uniquely as a list of ballots (hence there exist no partial ballots which could be combined in multiple ways, as for example it is the case in the ThreeBallots scheme) [82].

Eligibility Verifiability

> Anyone can verify that each vote in the published set of "all" votes was cast by an eligible voter, and anyone can verify that each eligible voter cast at most one vote [155].
>
> Anyone can check that each vote in the election outcome was cast by a registered voter and there is at most one vote per voter [162, 268].

Introduced was the property only as recent as 2010 [162, 268], although it is merely a rebranding of *Democracy* which has been known already 1992 [124, 123]. It captures the missing piece in the *chain-of-custody* definition of end-to-end verifiability: Voter authentication [127, 26]. Only the votes of eligible voters must be counted, and for each eligible voter only a single vote must be considered.

This property seems to be little researched in the literature, and few schemes explicitly claim it. Notable exceptions include an extension of Helios (2015) [167], and the Electryo scheme (2021) [252], with both implementing the property by simply publicly allocating each vote to a voter.

Accountability / Dispute Resolution

> Accountability is fulfilled when, for a judge J:
>
> - Fairness: J (almost) never blames protocol participants who are honest, i.e., run their honest program.
> - Completeness: If, in a run, some desired goal of the protocol is not met — due to the misbehaviour of one or more protocol participants — then J blames those participants who misbehaved, or at least some of them.
>
> [171] *(slightly paraphrased)*
>
> A voting system is said to have dispute resolution if, when there is a dispute between two participants regarding honest participation, a third party can correctly resolve the dispute [26].

Introduced in 2010, it is a stronger notion of verifiability: When something goes wrong, it is not only detected, but additionally it is established *who* misbehaved. *Global Verification*, where "the published result exactly corresponds to the votes cast by eligible voters", is shown to be a special case of

Accountability [171]. For practical schemes, the authors argue it a must-have [168].

The idea of *Accountability* was picked up by other authors, and extended to additionally require also some way to process the consequences. The resulting property is then called *Dispute Freeness* and guarantees disputes are settled (implying the protocol can continue afterwards) [26].

Due to its relative novelty, this concept has not received much attention yet. Approaches would likely include signatures (due to non-repudiation), but might also require help from the outside (for example, to counter a collection authority that simply drops all received votes). Formal analysis of the property suggests primarily processes concerning the individual verifiability checks may be difficult to protect from disputes [15].

3.1.3 General Security Properties

Some properties required for secure systems in general are described here. While not strictly e-voting specific, some of their aspects nonetheless have e-voting specific implications.

Availability For the voters to actually benefit from the system, it must be operational, both for the voters as well as all other actors of the system.

Users may interpret an unavailable system as breached. Additionally, an excessively slow system might hurt participation. Disruptions within the fragile internet architecture are feasible, especially for state-level adversaries [223]. Targeted disruptions (against specific voting times or locations) make the problem even more serious, as the election result could be skewed into a specific direction [26].

A related concept is robustness.

> Faulty behaviour of any reasonably sized coalition of participants can be tolerated. No coalition of voters can disrupt the election and any cheating voter will be detected [249].

This looks at availability from the protocol perspective: Malicious actors may disrupt the election by (ab)using the protocol specification. For example, a trustee might prevent threshold-decryption of the election result by refusing to share its partial key.

Usability

> (Usability is the) extent to which a system (...) can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [122].

In this definition from ISO, effectiveness requires accuracy and completeness when achieving specified goals, efficiency ensures the resources required are low, and satisfaction necessitates comfortable usage. Therefore, usability ensures the system is accessible in such a way that the user can actually benefit from its intended purpose and properties.

Similar to availability, usability must also be fair in such a way that it does not skew the result. For example, the system must be provided in all official languages to not hinder participation of communities which do speak only a subset of official languages.

An additional challenge is to help the voters understand the verifiability and privacy properties, to ensure their motivation following proper procedures. Especially verifiability has been found to not be easily understood [288, 187, 193, 210], similar results can be expected for the privacy properties.

Transparency When the system is transparent in its operation. Voters and auditors can reasonably follow what is going on and attest correct procedures. For this to be possible, all necessary information must be published in an accessible way.

We interpret this term in a very general way: It applies to specification, implementation, review and operation, but also to the surrounding processes such as regulation and organisation. If all people involved with the e-voting system would be replaced at the same time, and no introduction would be given to the newly commissioned, could they pull off an election like before, knowing *what* to do *when*, and *why*?

This motivates the second very important part required for transparency, besides publishing all information: The information must also be understood. While it cannot be expected that everyone understands all the details - for example, cryptographic proofs do require very specific expert knowledge - a ”reasonably high percentage” must understand ”sufficient”. While this is unfortunately very fuzzy: A simple system - with few actors, processes and cryptographic operations - clearly has an advantage over a more complicated one.

3.2 Notable Schemes

We review selected e-voting schemes that have been proposed in the literature or industry. We consult various surveys [155, 199, 220, 158, 166] and rely on our general overview of literature. We focus on schemes with novel ideas, real-world usage or lasting impact on the research community. However, we by no means claim completeness, specifically having not considered schemes which seem hardly applicable to the overall purpose of this work.

We do not consistently compare properties, nor explain the cryptographic building blocks. At this point of the work, we are only interested in reviewing the general ideas and mechanisms. We however align terminology somewhat, naming authorities consistency *Collection Authorities* (those who store the vote submitted by the voter), *Tallying Authorities* (those who tally the stored votes) and *Trustees* (those who jointly possess key material to decrypt the election result). Seldom, we also require *Registration Authorities* (those who register voters and provide each a credentials over a secure channel), *Printing Authorities* (those who print and send key material over the (secret) postal channel to voters) or *Authentication Authorities* (those who can authenticate individual voters)

3.2.1 Supervised Schemes

While we interest specifically in remote e-voting, much research has gone into how supervised voting can be improved. Voting machines or specially prepared paper ballots allow various kinds of interesting schemes with different privacy and verifiability tradeoffs. We look at these schemes as some of their central ideas may also be applicable to remote voting schemes.

What separates supervised schemes from remote schemes is that they are able to force a specific process on voters, due to the physical presence of the voter in the voting area. By restricting voter behaviour, some strong verifiability and privacy guarantees can be reached without using much cryptography. Consequently, porting the central ideas from the supervised schemes to the remote setting might not always be possible.

We separate here from (non-verifiable) direct-recording electronic voting machines (DREs). Their setup is usually dead simple (touchscreen allowing to compose the vote) as are their security mechanisms (some print a paper receipt for manual recounts, some have video surveillance, ...) [134, 29]. While they promise to bolster trust in the election [135, 224] and improve usability for voters [192], their quality is often lacking[17] and financial incentives might play a non-negligible role in vendor selection [71, 188]. As we interest in this work primarily to build a suitable internet voting for Switzerland - which has never used DREs so far, and there are not plans to do so - we consider a more detailed look at DREs as out of scope for this work.

We omit describing here several schemes we find nonetheless interesting. Some clever techniques were presented by Punchscan [110], Aperio [109] and its digital twin Eperio [111]. Primarily practical problems due to complex votes were tackled by STAR-Vote proposed for Travis County, Texas (US) [18] and EasyVote proposed for the State of Hesse (DE) [283].

[17] Many reports about broken DREs exist, for example https://politi.co/2K2OGOv/.

Prêt-à-voter family The voter receives a printed two-part paper ballot. On the left part, the candidates are printed in rows in some random order. On the right part, the voter can mark its preferred row. Further, the right part contains an encryption of the candidate order, which can be threshold decrypted by a set of trustees. Random audits ensure the order of the candidates on the left actually conforms to the encrypted order on the right.[18]

The voter now marks its preferred row on the right side and then destroys the left side with the printed candidates. The voter gets a copy of the right side as a receipt before submitting the original to the collection authority.

The collection authority publishes for all received ballots the encryption of the candidate order and which row was marked, for the voter to verify with its receipt. The trustees then use this information to decrypt the candidate order, and tally the votes [72].

Many variants have been proposed based on this simple design. They improve on generation of the ballots, reduce assumptions on authorities, introduce a human-readable paper trail or provide everlasting privacy [257, 253, 182, 170, 95]. A version of Prêt-à-voter called vVote [62, 90] was used in 2014 in Victoria (AU) in a binding election. It was available to visually impaired voters, voters not proficient in English and voters from abroad. A total of 1121 votes were handed in using the system [63]. However, the system did not convince [219] and was abandoned again [276].

The interesting idea from this voting scheme is, that not the voting choice is cryptographically processed, but rather the context in which it was given (concretely, the ballot sheet). While this is not cryptographically simpler per-se (still, verifiable shuffles or homomorphic encryptions are needed by the trustees to decrypt the vote while keeping verifiability and privacy guarantees [256]), from a voter perspective, this could be desirable: The vote is published in clear text which may makes the check whether the vote was collected as expected simpler and more intuitive for the voter.

Voters need to be supervised to ensure they do not get a full receipt[19] and to prevent the chain-voting attack[20].

Scantegrity family The voter receives a paper ballot with a unique id, the candidates, and an initially hidden code for each candidate. Random audits

[18] Some unused ballots would be randomly chosen, and the trustees would verify whether their print matches their encrypted candidate order.

[19] For example, by not destroying the left part.

[20] An attacker smuggles out a ballot, then coerces a voter to use this specific ballot to cast its vote (of which the attacker of course knows the candidate order). This attack can be chained by additionally coercing the voter to smuggle out the new unused ballot they received from the polling place.

ensure the hidden codes are correctly committed to their respective candidates and ballot ids. The voter uncovers the code for the favourite candidate (hidden behind a seal) and notes it down together with the ballot id. Then the voter passes the paper ballot to the collection authority.

The collection authority ensures only an appropriate number of candidate codes are unsealed, then scans the ballot, and publishes online pairs of ballot ids and their uncovered code. Voters check whether the correct uncovered code appears, else can challenge the authority (knowing the uncovered code is assumed sufficient proof). In the tallying phase, the uncovered codes are then matched back to the candidates based on previously committed assignments [70].

Scantegrity II was used in Takoma Park, Maryland (US) in 2009 to cast 1728 votes. It was used again in 2013 together with Audiotegrity and Remotegrity [64]. Audiotegrity adopted the Scantegrity protocol to visually impaired voters; essentially giving them the choice to either cast or audit a Scantegrity ballot produced by a computer [159]. Remotegrity adopted the Scantegrity protocol to remote voters, giving them the possibility to audit the vote online. The voter sends the code of the favourite candidate to a public bulletin board, then checks the board updated accordingly. If this is the case, the voter unseals the authentication code and sends it to the bulletin board, too.[21] However, the vote still needed to be sent in to the authorities to produce a paper trail. Two out of the five voters using the system forgot this and invalidated their vote [287]. After 2013, Scantegrity was again abandoned [64]. Before Scantegrity, some of the authors were involved in the Punchscan system, used in 2007 in student elections [110].

The scheme only requires a minimal change in voter behaviour: The voter no longer places a cross besides a candidate but instead uncovers the candidate code. This already enables an end-to-end verifiable voting system, notably including dispute-freeness, as the voter can successfully challenge an authority that publishes the candidate code incorrectly.

The core mechanism is the destruction of seal behind chosen options, to make the choice permanent (and the non-choice provable). Knowing what is behind the seal, combined with a check only an appropriate number of seals were uncovered, is enough proof. To execute this check, Scantegrity needs voter supervision.

Remotegrity uses the seals the other way around to not require vote supervision: By showing an untampered seal, the voter proves they have not confirmed their vote. This however is no longer dispute-free, as the voter

[21] Note that the voter will not unseal if the bulletin board is incorrect. If an authority confirms some voter's invalid vote, then the voter can successfully challenge by showing its still sealed confirmation code.

cannot prove whether they attempted to vote. Additionally using seals for the candidate codes does not strengthen this property, as the voter can still not prove that they have actually sent the codes to the server.

ThreeBallots The voter receives a three-part paper ballot. Each part has its own unique id and a full list of all candidates.

For each list having n candidates, the voter places $n + 1$ marks as follows: Each candidate is marked on either one of the three parts, while the favorite candidate is marked on either two of the three parts. The voter chooses one of the three parts, and gets to keep a copy of it as a receipt. Then all three parts are submitted to the collection authority (which does not learn which part was kept as a receipt) and mixed with the parts of other voters.

The collection authority publishes all parts of all voters. Now any voter can verify the part of which they own a receipt for is actually included in the tally, and anyone can tally which candidate won.

The scheme achieves receipt freeness as the voter, with the receipt of a single part, cannot convince anyone for how it voted.[22] Still, the collection authority dropping parts is discovered with probability of at least $\frac{1}{3}$.

The scheme uses no cryptography at all, it requires however supervising the voter to prevent them getting a full receipt[23]. Further note that the receipt freeness only holds with few candidates and many voters (else coercing voters may agree on identifiable candidate marking patterns) [245].

3.2.2 Remote Schemes

Remote voting enables to vote from home, or in fact, anywhere the voter pleases. While convenient, or even the only way the vote can be cast (for example, for voters living abroad), additional difficult security challenges arise.

Coercion of the voter becomes easier for the attacker. The voter no longer enters a private voting booth, but the attacker can literally watch over the voter's shoulder while casting its vote. Further, the vote has to be cast on the voters device, then sent over the internet, before it is digitally collected and tallied. The voter cannot reasonably trust any of the involved devices, and needs proof which verifies the vote has been cast, collected and tallied as expected. Finally, the voter has to go through all these hurdles self-reliant. And, from the point of view of the voting authorities, without them being able to force some kind of process on the voter.

[22] This is why at least three ballots are required: With only two ballots, the voter could not vote for its favourite candidate, but take home a receipt where the candidate is not marked.

[23] For example, by taking home a copy of more than one receipt.

We focus here on a specific form of remote voting, namely where the vote is transferred over the internet to its intended destination. While improvements over postal voting would also be interesting and important to look into, we focus on internet voting as we feel this is a more pressing issue.

We omit describing here several schemes we nonetheless find interesting. In Switzerland, UniVote and related schemes were verifiable schemes used in student elections [140, 103, 102]. The same researchers have also proposed more theoretical schemes which fulfil very strong coercion resistance and everlasting privacy properties [177, 180, 178]. Very interesting from a formal perspective are fully verified proposals [169, 14, 141]. Also especially interesting in the context of this work are code-voting based schemes [68, 151, 154, 153, 28, 201].

FOO scheme The voter prepares its vote and blinds it. The blinding ensures no conclusions about the plain vote can be drawn. Then, the voter sends this blinded vote to the authentication authority, which ensures the voter is eligible to vote and responds with a signature over the blinded vote. The voter unblinds the vote, and sends it together with the - still valid - signature to the collection authority through an anonymous channel.

The collection authority ensures the signature is valid, then includes the vote into the public tally [124].

With the sole assumption of non-colluding authentication and collection authorities a rather simple scheme results. Note that the voter can even proof when the registration or the collection authority misbehaves. A drawback however is the difficult practical implementation of the anonymous channel [92] from the voter to the collection authority. Notably, when we opt for an implementation using a verifiable shuffle, then we can design a verifiable voting system without blind signatures at all [220].

Helios family The voter selects their favourite candidates, then the client-side application uses a randomized encryption to create the vote. The application commits to this vote (for example, by displaying a hash of its ciphertext) and leaves the voter two choices: Either audit the vote or cast it to the collection authority. When the voter chooses to audit, the application displays the randomness used for encryption, so that the voter can verify its candidates were correctly encrypted into the vote. When the voter instead chooses to submit, the vote is transferred to the collection authority without the randomness revealed [19].

The collection authority publishes all received votes and who cast them. After a cryptographic shuffle (to separate votes from voters), the votes are decrypted and counted [2].

As with other successful schemes, many variants have been proposed; to use verifiable shuffles or threshold encryption, defend against ballot stuffing or provide everlasting privacy [221]. The security has also been improved; for example a replay attack has been discovered which could endanger vote privacy [86]. In the context of Helios, the important difference between weak and strong Fiat-Shamir[24] has been studied [25]. A variant of Helios called KTV-Helios provides private eligibility verification: Anyone can verify only eligible eligible voters have submitted votes into the tally, while it remains unclear which vote was actually submitted due to the use of dummy votes [167]. Another strain of Helios using weaker trust assumptions while providing eligibility verifiability and receipt freeness is the Belenios family [83, 66].

Helios has been available online since 2008 [25], and has been used in university elections [3], the IACR[26] and others. This "official" implementation has been continuously updated, like the switch of the shuffle in Helios 1.0 to the simpler and more efficient homomorphic tally in Helios 2.0 [221]. The Belenios implementation is available since 2015 and has also undergone several revisions since its release.[27]

The essential mechanism proposed of giving the use the option to either audit the vote or cast is called the Benaloh challenge in the literature. It tries to ensure the voter's untrusted device is forced to perform correct computations: As it has to commit to the result of its computation, and said computation might be checked by the voter, wrong computations risk detection.[28] However, if the computation is complex (as it is the case with Helios), verifying it might be challenging for the user: It may require a second device and transferring the potentially long data required for verification onto this. Little intuitive is also the fact that the audited vote cannot be submitted afterwards, but rather a new encryption has to be requested (our informed reader of course understands that this is required for receipt freeness) [187].

JCJ/Civitas family The voter receives a private credential from the registration authority. During the voting phase, the voter encrypts the credential as well as its preferred candidates, and submits both to the collection au-

[24] For zero-knowledge proofs of knowledge (ZKPK), to prove a statement the prover commits to some value, then the verifier provides a challenge, to which the prover then answers based on the commitment. If one would like a non-interactive ZKPK (hence without the prover interacting with the verifier), this challenge is instead generated by hashing the commitment (weak variant) or by hashing the commitment and the statement (strong variant).

[25] Reachable under https://vote.heliosvoting.org/.

[26] International Association for Cryptologic Research. The first vote using Helios was conducted in 2010, see https://www.iacr.org/elections/2010/.

[27] Reachable under https://www.belenios.org/.

[28] Albeit this might be very hard to detect. How could the voter distinguish phony from real randomness used for the encryption?

thority. Voters also have access to forged credentials which they may use to submit additional votes.

The collection authority publishes all received votes. After shuffling them, and therefore anonymizing them, votes with forged credentials are filtered out: The registration authority publishes a list of all valid encrypted credentials, and using plaintext equivalence tests, the tallying authority discards all votes with forged credentials. The remaining votes are decrypted and counted [75].

Civitas' predecessor was the JCJ scheme, proposed to tackle coercion resistance as introduced in the same paper [157]. Civitas keeps the concept of encrypted credentials such that real and forged credentials are indistinguishable, but relaxes the trust assumptions on single involved authorities (for example, Civitas mandates multiple registration authorities instead of just one) [75]. Further improvements to relax trust assumptions on authorities have been proposed [266].

The essential mechanism used to achieve the coercion resistance relies on valid and forged credentials, with the difference unobservable to the attacker. The implementation of this idea as done by JCJ/Civitas requires complex cryptography and multiple non-colluding authorities. The approach in general has further theoretical and practical drawbacks. As the votes with forged credentials are only filtered out after the shuffling, the voter has no way to assert its intended vote was actually tallied, arguably breaking cast-as-intended [158]. Initial performance of tallying was low, although this was improved by other researchers [271]. The amount of forged credentials a voter can produce must be bounded, to avoid flooding the bulletin board with invalid votes [10, 160]. The handling of many credentials by the voter might require assistance, for example using Civitas-specific smartcards, a non-trivial expense for authorities [200, 202]. Further, the private credentials must be exchanged on a secure channel without leaving any evidence, likely requiring physical presence of the voter at some safe location.

Pretty Good Democracy Pre-election, $(k+n)*(m+1)$ random codes are generated, for k audits, n voters, and m candidates (the $+1$ is used for the acknowledgment code). These are encrypted under the trustees public key and then verifiably shuffled into the table P with $k+n$ rows and $m+1$ columns. The trustees help decrypt P for the printing authority, which sends each voter the content of one row and the corresponding row index, and uses the remaining k rows to audit the codes. Each trustee then shuffles the first m entries of each row, storing the permutation homomorphically in an additional value per row, resulting in the table Q (with n rows and $m+2$ columns). Q is published on the bulletin board.

The voter selects the code associated to their favourite candidate and sends it together with the row index to the collection authority. The collection authority encrypts the code and posts it on the bulletin board next to the correct row of Q. The trustees run plaintext equivalence tests to find which column index matches the encrypted code and mark the specific entry of Q. Then, the trustees collectively decrypt the acknowledgment code and return it to the voter.

After voting and for each row of Q, the trustees encrypt the index of the marked column and homomorphically add it to the encrypted permutation. These final values are decrypted, and determine the chosen candidate for each row [258].

The security relies on non-colluding trustees as well as non-colluding printing and collection authorities. Colluding printing and collection authorities can break both privacy (undetectable) as well as vote on voter's behalf (detectable). This strong trust can be reduced with distributed printing (albeit not that practical) [254]. Through the initial shuffle resulting in P, the scheme needs to place no trust in whoever generated the voting codes: The code-candidate assignment is kept private and some of the codes intended for the voter are audited.[29]

The main contribution of the scheme is the combination of code voting combined with receipt freeness[30], through encryption of the vote. No trust on the voter device is required: Privacy is ensured through code voting, and integrity (correct encryption of the vote) through the acknowledgment code (which the trustees only publish when they ensured the vote is indeed an encryption of one of the codes associated to the voter). However, individual verification is not provided, as the returned acknowledgement code is not tied to a specific vote (rather than just to *some* vote).

Selene family The voter encrypts its vote, and sends it to the collection authority. The collection authority pairs the encrypted vote with an encrypted tracking number and stores it on the public bulletin board. It then replies with a commitment to the tracking number as a receipt for the voter. The voter knows a trapdoor for this commitment, hence can open it to any tracking number they desire. After the election, the tallying authority shuffles and then decrypts votes and their associated tracking numbers. All pairs are posted on the public bulletin board.

[29] From a theoretical point of view, this does not implicate the security of the scheme: We could instead task the printing authority with creating P, without any changes to our trust assumptions. However, from a practical point of view, the additional introduced complexity might be worth it: Generating much random data can be difficult, and may not be realistically doable by an authority specialized on printing.

[30] Code voting avoids placing trust in the voter device for vote secrecy, while receipt freeness limits opportunities for coercion.

The voter, if coerced, can now choose a vote it wants to claim ownership, and can fake a receipt using its trapdoor. Some time later, the tallying authority opens the commitment to the voter, so they can check that their vote was included correctly. Attackers cannot distinguish real from fake receipts, hence why receipt freeness is preserved [255].

This mechanism can also be applied to other schemes [252] while the usability has not been conclusively demonstrated [8, 288, 166]. Its implementation and operation is complex: It requires verifiable shuffles, trapdoor commitments and other complex cryptographic primitives, and - practically likely the biggest problem - a private channel from the tallying authority to the voter after each election.

The main idea is that trapdoor commitments are applied to internet voting: The voter sends the ballot to the election authority which uses a trapdoor commitment scheme to commit to its reception. The election authority can only open the commitment to the specific vote, hence can convince the voter it was included correctly. But as the voter knows the trapdoor, it can fake a receipt to any vote, hence is not able to convince a third party how they voted.

3.2.3 Industry protocols

Remarkably, industry never used directly the ideas from research, but instead always proposed and implemented their own protocols. This could however just be coincidence as the sample size is low: We identified only two strains of protocols in use. One is made in and for Estonia, the other is made in Spain by Scytl. Scytl remains to be the only international firm providing internet votings for governments.

Estonia introduced internet voting in 2005, already at a national scale [184]. In almost yearly elections (municipal, national and european) internet voting gained more an more users [185]. In 2013, individual verifiability was added assuming a non-colluding secondary device [150]. In 2017, auditability of the server processes were improved, and based on this the system was argued end-to-end verifiable [148, 145]. Further hardening the system was again requested politically in 2019, with some concrete improvements proposed in 2021 [145]. In general, the system is seen as an overwhelming success, making voting safer, cheaper and more convenient [282, 164, 163].[31]

Scytl provided internet voting systems since 2005 [152]. Their first publicly reviewed system was for the Norwegian elections 2011, with the protocol published at a research venue [6, 176]. The essential (individual) verifiability

[31] However, it should be noted substantial critique has been raised towards over reliance on the voter's device (shortly before introducing individual verifiability) [146] and operation procedures (before introducing universal verifiability) [270].

mechanism was based on return codes, as explicitly chosen by the Norwegian government [176], although it was not prominent in the literature at the time. Since then, the essential idea behind the Scytl protocols stayed the same, while over time more cryptographic operations were added to reach stronger security guarantees.[32]

After the first contract in 2005 (since its foundation in 2001), Scytl quickly grew. Besides internet voting, they provide voting services of various sorts, including DREs [152]. The steep rise to 600 employees by 2015 in only a few years [152] was followed by a evenly steep decent into bankruptcy in 2020 [173] and the acquisition by the Paragon group for only 5 million dollars [218]. Whenever their systems were revealed to the public, controversies arose: Almost matching experiences about a lack of implementation quality were reported in their 2011 and 2019 reviews in Norway and Switzerland, respectively [217, 179]. Their systems were cancelled a few days before the vote as in Sweden 2019 [104], or abandoned after public support eroded as in Norway 2013 [31] or France 2017 [243]. Only New South Wales (AU) is still providing their internet voting system, where strong legal provisions prohibit inspecting the system closely [276].

Estonia protocol Estonia started plans to introduce internet voting as early as 2002 [282]. In 2003, a protocol was proposed primarily focusing on simplicity, using the national electronic ID system for authentication and placing full trust in central servers [146]. The public tender 2004 won Cybernetica, the company still involved with further developing the system [185], which already contributed to the national electronic ID system.[33] The first trial was held at the start of 2005, and the first nation-wide elections already at the end of the same year. The main motivation was to increase voter turnout, although whether this was achieved was hard to tell, in part because only 2% used the internet voting channel in that first election [184].

This first generation system used a simple "double-envelope" mechanism [184]: The vote is encrypted[34], then signed by the voter using the national electronic ID.[35] Two online components are in operation: The first authenticates the voters, while the second stores the votes. The offline component receives all votes after the voting phase, stripped of their signature to pre-

[32]Compare for example the Norwegian system 2011 with the Swiss system 2019: The cryptographic primitives are the same, as is the essential idea. However, the Swiss system additionally achieves universal verifiability, essentially by attaching zero-knowledge proofs to all computations (we simplify a bit, but overall this is the approach taken).

[33]See https://e-estonia.com/e-estonia-podcast-cybernetica/.

[34]Given an RSA election public key and a random seed r using RSA-OAEP.

[35]Signatures using that ID are in Estonia legally equivalent to hand signatures [183]. Malicious actors with access to voters ID could compromise other (arguably more valuable) systems such as loan granting procedures.

vent voter identification, to decrypt and tally [146].[36] Both the application the voter used to hand in the vote, as well as the involved servers, have been hardened, but are nonetheless fully trusted in order to archieve correctness of the system [146]. Security was thus mainly achieved using classical risk analysis and mitigation by organisational measures. To improve vote privacy, revoting over the internet was allowed[37], with the postal vote additionally overriding any internet vote [184].

In the 2007 national elections, the system was used for the second time, tripling the number of votes cast to 31k [183]. This trend would continue, with each election more and more voters choosing the internet channel [185]. Transparency efforts were increased; for example a public test system was provided and a national information campaign done. During the vote, the current number of internet votes cast was published, allowing a sanity check whether the number is reasonable [183]. An increase in overall voter turnout is not observed, although it is argued that the new voting channel manages to include voters which otherwise would not engage politically [281].

In 2011, the fifth time using the first generation system, the strong trust assumptions on the voting client sparked its first controversy: A modified voting client was presented to the media (displaying different actions than those actually executed), leading to a (denied) appeal to invalidate all internet votes [146].

Until the next elections in 2013, after a long period of yearly elections, politics and implementation caught up to the inputs from the expert community [282]. Following this discussion, the system was extended to support individual verification and made open-source [185].

Individual verification was added on top of the already existing system, and is again conceptually very simple. After casting the vote, the voter now receives a voting reference. The voting reference and the random seed r used for encryption of the vote are transferred to a secondary device (like a smartphone) using a QR code. Then, the device downloads the vote using the voting reference. It encrypts all possible voting options using r[38], and checks which one of the resulting ciphertexts corresponds to the downloaded vote. The corresponding plain value is then shown to the voter [150].

The design explicitly avoids that the device learns the plain vote the voter expects, so it cannot simply fake validation. Further, the smartphone apps run on different operating systems as the voting application, hence forcing the voter to use different devices. Arguably, coercion resistance does

[36] The offline component only receives the last internet vote of each voter, respectively no internet vote if the voter has used the postal channel [146].

[37] The voter is notified if their to-be-cast internet vote overrides an existing vote [146].

[38] This was feasible due to the low number of voting options.

not decrease, as revoting remains possible as before. It is however acknowledged that the voting application could show the voting reference of another voter's vote (and then submit any vote it likes). This threat was mitigated by making voting references valid only for a short time period (30min - 60min) and limiting how often they can be retrieved from the server (3 times) [150].

In 2013, no problems with verification were reported. In 2014, few problems, which could be traced to bugs in the verification apps, were reported. Overall, the high verification rate of 3-4%, with no real issues detected, results in a very low probability of large-scale attacks [150]. Log analysis attested the new system overall very stable operation, with most of the few anomalies conclusively explained [149].

However, new trouble arose with the strong trust assumption in the server-side components and national ID system. Researchers from the US observed the voting procedure of 2013. They identified procedures followed inadequately (like executing four-eye-principle tasks by a single user, skipping consistency checks or generally improvising in case of unexpected failures) and lax operations of the servers (like downloading software over HTTP or logging in as root). They demonstrated a range of attacks, however each attack relied on broken trust assumptions either on the voter side[39] or the trusted server side[40] [270]. The paper concludes with a recommendation to abandon the internet voting channel.[41] The Estonia Election Committee rejected the recommendation, claimed the described attacks as unfeasible and regretted the circumstances of its publication[42] [80].

In 2016, further development of the system was initiated to relax the strong trust assumptions on the server. The largest threats identified for the current system were that votes could be removed from the collection server, and that the tally had no verification mechanisms so far. The new system called IVXV introduces the new server component *Registration Service* (RS), which would work in tandem with the collection server to prevent tampering with the recorded votes (essentially both keeping a ledger of all collected votes). Further, the RSA-OAEP encryption of the votes is replaced with ElGamal, enabling privacy-preserving verifiable shuffling and proofs of correct decryption [148].

IVXV slightly extends the voting process: After voting, the voter receives, additional to the voting reference, a signature of the RS over a hash of the

[39] The *Ghost Click Attack* revotes by having access to the ID and the *Bad Verify Attack* assumes the secondary device colludes with the vote casting device.

[40] Various forms of malware on the servers and their potential impact were demonstrated.

[41] The tone of the paper is rather alarming and sensational, with a corresponding webpage published https://estoniaevoting.org/. The paper remains widely cited.

[42] The webpage disclosing the findings went online less than week before the next election. As described in [270], election officials suspected a political plot behind its creation.

collected vote and the voter identity. This signature is checked by the secondary device, which notably now learns the voter identity during verification. The security of the system is hardened as an attacker would now need to compromise both the collection server as well as the RS. It was also argued as an advantage of the new design that the collection server, as not fully trusted any more, could now be operated by an external provider [148].

The tallying is extended to add proofs over correct decryption. All other processes, notably from the tallying component, can be audited by executing the same algorithms over the same data. Auditing however reveals which voters have revoted and even the vote - voter relation. The latter could be avoided by introducing a verifiable shuffle.[43] Further, effectively auditing requires expert knowledge of the underlying cryptographic protocol and likely even an independent implementation. It was thus proposed that trustworthy auditors would be appointed by specific stakeholders (like political parties) rather than making auditing open to the general public [148].

With these changes, the system is argued end-to-end verifiable [148]. It was used for the first time in 2017 [145]. In 2019, political efforts were again made to continue developing the system. Besides overall hardening the system, it should also be examined whether voting from mobile devices is feasible. So far no conclusion has been reached, but small changes have already been proposed, for example an independent notification channel to the voter (confirming whenever a vote has been cast). It is made clear that the system will continue to undergo changes [145].

Using the published logs, over the years some conclusions about internet voting behaviour was drawn. Some voting in pairs is observed, both parent-child as well as between spouses, indicating the secrecy of the vote becoming more of a (free) choice to voters [279]. It was further found that verification indeed improved confidence in the system, and verification was executed predominately by the more risk-aware population[44] [269].

In Estonia, the internet voting system (and their digitalized government in general) is clearly seen as a wide success, both as an indispensable service to its voters, and also a source of national pride. From an efficiency point of view, internet voting also prevails: It was both in 2017 as well as in 2019 the most cost-effective voting channel, with the bulk of its resources spent on operations [164, 163].

Scytl in Norway (2011-2013) In 2008, Norway started the e-valg project, which aimed to introduce internet voting in the municipal elections in 2011.

[43] The verifiable shuffle was declared optional (depending on the degree auditing would be open to the public), and it remains unclear whether it was considered in the final implementation.

[44] Identified as male, 18-40 years old, Linux users.

In 2009, the core strategic decisions were made: A single central system was aimed for, using an eID to authenticate the voters. It was to be tried in some selected municipalities. A dedicated team of around 10 people supervised the project, while experts, users, politicians and administrations were included in feedback groups. The whole cost of the project was estimated at 123 million kr (around 12 million EUR), with an 85% chance to stay under 220 million kr [203, 205]. A public tender was held in 2009, for the system to be implemented in 2010 and afterwards tested [204].

The protocols were to use *return-code voting*[45]: The voter enters the vote in cleartext, sends it to the server (possibly encrypted) and receives back return codes. The voter ensures these codes match with what was previously received over a secure channel (e.g. postal mail). Compared to the code voting approach, the voter does not need to input codes for their selection, but only needs to compare, which is argued to be much easier [6]. However, the voter now must trust their voting device in order to achieve privacy, as the vote is entered in clear text. The Norwegian authorities "strongly felt" that the usability advantages of return-code voting are more important than the security advantages of code voting [176].[46]

In the implemented municipality elections, the voter chooses a party, then adds specific candidates to the list of the party. The vote therefore consists out of a single party selection, plus up to 99 candidates, while no write-ins are possible. Revoting is allowed, while the last submitted ballot is counted. Voters can override a remote vote using postal mail voting. Universal verifiability is argued impractical as traditionally, only very few information about the count is published, and as many different ballots are possible with the same effect on the count, Italian attacks would complicate implementation [132].[47]

The tender received two submissions [176]. One submission pioneered the oblivious-transfer idea later on used in CHVote, submitted partially by members of the same firm that also developed the Estonia voting system [147]. However, their proposal was prohibitively inefficient, while still requiring non-collusion assumptions on the two involved servers for both correctness and privacy [176]. The winning bid by Scytl was much more efficient with arguably weaker security assumptions, although it required an additional trusted setup phase [132].

[45] Sometimes also referred to verification code voting.

[46] We could not reconstruct the reasoning leading to that conclusion. The provided security guarantees are different for code voting and return-code voting, as is the mental model important for usability, hence a one-on-one comparison may compare apples to oranges.

[47] However, some form of verified count was implemented in the tallying procedure, although only auditors of the government could check this [273]. The government saw verifiability also as a useful tool to ensure correctness of their system, besides being a necessary evil to get support of academia and for the system [128].

The protocol was initially designed by Scytl [6, 176], then modified by Gjøsteen which also performed a computational proof [132, 133, 7]. This modification was formally proven by Cortier et al. in two stages, 2012 and 2016 [87, 88]. The final implemented protocol was again different, primarily to improve performance by moving operations from the voting client to the servers [7]. However, the core ideas - to rely on two non-colluding servers and how return-code voting was implemented - stayed the same throughout all proposals. We will only describe the last approach, as this one was actually used during the election in 2011 [7] and then continued to be used in future Scytl protocols.

First, two public/private keypairs are generated: One for the vote collector server (VCS) and one for the return code generator (RCG). Shares of the private keys are distributed among electoral board members. The election public key is generated out of the two public keys of VCS and RCG.[48]. The election private key is not constructed yet, but will be after the election using the shares of the electoral board members. Throughout the whole protocol, it is assumed that VCS and RCG do not collude for vote privacy. Intuitively, VCS stores the vote identities, and RCG the return codes. Further, VCS and RCG must not collude for integrity, as they could freely add, alter or remove votes.

For each voter V_i, VCS applies a pseudo-random function[49] to get the voter secret s. All possible selections of the voter are raised to the power s to result in the partial selection codes SP_{ij}. All SP_{ij} are passed to the RCG together with V_i. The RCG applies another pseudo-random function[50] over each of the partial selection codes to result in the final selection codes S_{ij}. For each S_{ij}, a return code is chosen, and then a hash table is built up: The key is given by $H(S_{ij})$, while its value is an encryption of the corresponding return code, using S_{ij} as an encryption key. Finally, VCS and RCG share their private keys and the hash table with the printing authority, which constructs for each voter a voting card with all selections and the respective return codes.

In the voting phase, the voter makes their selection and encrypts it using the election public key. The vote is sent to VCS which stores it for tallying. Then, VCS partially decrypts it using its private key, re-encrypts it using s, then forwards the resulting value to RCG.[51] RCG further partially decrypts it using its own private key.[52] RCG can now repeat what has been done in the setup phase to retrieve the corresponding return code. The code is sent to the voter, who ensures it is valid [7].

[48] As ElGamal is used, this is easily possible due to its homomorphic properties.

[49] AES in CBC mode with an VCP-private key K_{VCS}: $s = \mathsf{AES\text{-}CBC}(K_{VCS}, V_i)$.

[50] HMAC with an RCG-private key K_{RCG}: $S_{ij} = \mathsf{HMAC}(K_{RCG}, SP_{ij}||V_i)$.

[51] With Zero-Knowledge proofs for RCG that the two operations were performed correctly.

[52] Note that by now, the election public key encryption has been fully reversed, while the vote is still raised to the power of s: Neither VCS nor RCG learn the voters intention.

To tally, VCS and RCG send their encrypted votes to the tallying service. Consistency is checked, and votes from voter which have also submitted a postal vote are removed. Then the votes are shuffled, which is verified by an auditor. Finally, the election private key is constructed out of the private key shares of the members of the electoral board. With this key, the votes are decrypted and counted [273].[53]

Out of usability concerns, and limitations of the used SMS channel to deliver the return codes to the voter in the voting phase, the return codes had to be much shorter than what is stored in the hash table. Consequently, another mapping was defined at RCG which mapped the long return codes to short 4-digit numbers. Further, not for all voter selections return codes were issued: Instead of every party and every candidate having its own return code (which would have resulted in a few thousand codes), in the election only each of the around 25 parties got its own code [7].[54]

In previous specifications of the protocol, RCG additionally returns a signature of seen ballots to VCS. VCS verifies and stores that signature, then sends it to the voter as a receipt [132, 133, 6]. It is not described why this part of the protocol was not implemented. It would have improved accountability, as in case VCS and RCG do not agree on collected ballots, the existing or non-existing signature would clearly place blame. But returning the receipt to the voter would have reduced privacy, as the voter can now prove how they voted.[55]

Entirely missing in the specification is how revoting is tackled. In previous specifications, a sequence number was determined by VCS and attached to the vote; then only the vote with the highest sequence number would have been counted [133].

Operated was the system by the government, with the components all developed by Scytl. VCS and RCG were run by different departments in different locations, as was the system that implemented the tally [273, 176]. The voter authentication was done over MinID (the Norway eID system), which required the voter to additionally place trust in their system to avoid ballot stuffing and similar attacks [273]. An additional component, besides VCS and RCG, was introduced to do the setup (up and until the printing authority step) [206]. Further detailed architecture of all systems were published, and most of it is preserved (although not the source code) [206].

[53] We explicitly note that this description is provided by authors not directly involved in the project, and no description of the tallying procedure is provided in the paper which describes the actually deployed version of the protocol [133].

[54] This usability / security tradeoff was deemed appropriate as 98% of voters only select a party list. The paper incorrectly claims on page 5 to additionally use codes for candidate positions, but clarifies on the last page this is not the case [7].

[55] The encryption of the vote is not randomized, hence the voter can prove its content.

When printing the voting cards, it had to be ensured the return codes were not attributable to identified voters (else, an attacker in control of the SMS channel could break vote privacy). To ensure not even the printer learns this association, printing was done in two isolated stages. First, the voting card was printed, folded and sealed. Only the voter id (which is required to identify the return code set by the VCS) was still exposed. Second, in a physically separate area, some eligible voter was bound to that id and the voting card appropriately shipped. The binding was wrongly attributed in rare cases [128].

In 2011, over ten municipalities 170k voters were eligible to vote [128]. 25% of the votes cast were over the internet voting system. The transparency efforts of the government were received positively, although operation details of the system were missing, and external audits and certifications were forgone [215]. The return codes were found to not be receipts, and hence compatible with the Council of Europe's *Recommendations on E-Voting* [13]. However, researchers found numerous bugs using off-the-shelf analysis tools in the source code after the 2011 election. Besides being hard to compile, the code base is huge (160k LoC) and was perceived as complex and generally of low quality [217]. This feedback would be largely repeated in the 2019 review of the same company's source code for the system in Switzerland [45]. The SMS channel was seen as inappropriate to deliver the return codes due to reliability issues of both GSM and the voters phone, which enabled undetectable revoting attacks [161].

In 2013, followed after a "contentious" debate, a slightly larger electorate of 12 municipalities with 250k voters was chosen for the trials [213]. Changes include a simpler printing process, a public bulletin board with hashes of submitted ballots[56] and some technical changes such as the switch from Java to JavaScript [31]. Due to a bug in the JavaScript implementation, 29k ballots were submitted with very weak encryption [213]. The bug was criticised as easily detectable with even rudimentary testing or code review [31]. Even though this was a critical failure of the system, it was not picked up much by local media [31].

After the vote of 2013, the newly elected government immediately stopped all projects related to internet voting [31]. In 2016 and 2018, in some municipalities internet voting was used again in local referendum[57] [30], but for elections no country-wide efforts were made again.

[56] Hence likely the proposal in early specifications, that RCG should send a receipt of the vote to VCS, with said receipt also shared with the voter, was finally implemented.

[57] See http://www.evalg2016.com/, deployed by the team at http://www.nvtc.no/.

Scytl in Neuchâtel (2015) Already back in 2005, Neuchâtel provided over its Guichet unique[58] internet voting for the first time [121, 55]. This system was already built by Scytl, in fact Neuchâtel was their first customer ever [152]. Little is known about the used protocol, it was before verifiability and other transparency measures were considered important in Switzerland [56].

In 2013, Neuchâtel extended the system to support federal elections [121]. In 2015, the system was further extended to incorporate an individual verifiability mechanism, likely to keep up with Swiss federal law requiring verifiability for larger electorates.[59] The cast-as-intended mechanism works similar as what was provided by Scytl to Norway 2011[60], but revoting is not allowed[61] and some cryptographic operations are moved to the client [126].

First, the election public/private keypair is generated, with its shares distributed among trustees. Also, a signature keypair is generated. Then, the code generation key K is created, which fixes a pseudo-random function F.[62]

For each voter V, a public/private keypair pk/sk and a confirmation value CV is chosen. The secret sk determines a random function f.[63] The finalization code is given by $FC = F(CV^{sk})$, and the return codes for each candidate c_i are given by $RC_i = f(c_i)$. Finally, references values $RF_i = H(RC_i)$ using a hash function H are determined, and published together with pk and a signature over FC on the bulletin board.

To end the setup phase, the servers are initialized: The vote collector server (VCS) is provided with the election public key, and the return code generator (RCG) receives K. Both receive read access to the bulletin board, while the VCS can additionally append.

The voter receives a voting card with its keypair, CV, FC and all RC associated to the respective candidates. The voter now selects its favourite candidates on the voting device. The candidates (represented as primes) are multiplied together and then encrypted with ElGamal using the election public key. Further, each chosen candidate is raised to the power sk resulting in the list l. Zero-knowledge proofs are generated to prove the randomness of the vote encryption is known, and that the encrypted vote contains the

[58] A "one-stop counter" for all civilian affairs towards the government.

[59] Individual verifiability was a necessary precondition if more than 30% (and universal verifiability if more than 50%) of voters were eligible [35].

[60] We explicitly state Norway 2011, as the RCG receipt allegedly used in 2013 [31] is not part of the protocol any more [126].

[61] As mandated by federal law [35].

[62] Concretely, HMAC is used as a pseudo-random function, and K is used as the key.

[63] Concretely, HMAC is used as a pseudo-random function, and sk is used as the key.

same candidates as the list l.[64] The proofs, the encrypted vote and the list l are sent to VCS.

The VCS ensures this is the first ballot cast by the voter, verifies the proofs and stores the vote on the bulletin board. The RCG is notified of the update, and uses l to construct the return codes as done in the setup phase. After checking each return code has a corresponding reference value, the return codes are forwarded to the VCS. The VCS relies the return codes to the voter.

The voter ensures the return codes match to what is printed on the voting card, and enters CV. The voting device computes $CM = CV^{sk}$ and sends this to the VCS, which forwards it to the RCG. The RCG verifies that $FC = F(CM)$ (using the signature published on the bulletin board), and then sends FC back to the VCS. The VCS stores FC on the bulletin board, hence marking the vote valid for tallying. The VCS further returns FC to the voter, which again ensures it matches with what is printed on their voting card.

After the voting period, the tallying authority collects all encrypted votes with valid finalization codes and shuffles them. The election private key is reconstructed out of the shares of the trustees and the votes are decrypted. As each candidate is represented by a prime number, the vote can be factorized uniquely, and counted appropriately [126].

The protocol is extended with a "usability layer" to shorten the length of cryptography material compared by or typed in by the user. The long return codes and the finalization code are mapped to shorter equivalents at the VCS.[65] Additionally, the user types in a PIN instead of its full private key, and instead downloads the private key from the voting server[66] [126].

As the private key of the voter is known by the voting server, clearly the voter has to trust it for vote privacy, and to not cast a vote on their behalf.[67] If the server casts a vote, the voter might detect such an abuse when they attempt to cast their own vote. Further, the server-submitted vote would not be counted as the finalization code is not attached (which can only be reconstructed once the voter enters their confirmation code). However, the secrecy assumption of the finalization code relies on an abuse of the signature primitive: A signature of each finalization code is published on the bulletin

[64] This is implemented by raising the encrypted vote to the power sk (this relationship is proved, too), and then proving that the re-encrypted vote is the same as the multiplication of the values in l.

[65] Each long representation C is mapped to its short equivalent sC using a key-value store structured like $[H(C), E(C, sC)]$ for E AES-128 encryption and H SHA-256 hash.

[66] It is not described whether the VCS or a separate component delivers the key. We assume in the following the VCS provides this functionality, as an entirely separate component would have likely been mentioned in the protocol description.

[67] In later proposals, the private key is instead contained in a key-store, unlocked by some voter-held secret [263]. We do not know if this mechanism was already in use here.

board, and it is therefore implicitly assumed, that the signature does not reveal what was signed. According to the protocol specification, RSA-FDH is used as a signature scheme [126], hence this assumption holds, but only due to the hash function used inside RSA-FDH. Explicitly using a keyed[68] pre-image resistant hash function as the primitive for this task would have been more appropriate. It should be noted the protocol description does not discuss the security implications of the "usability layer" [126].

Compared to the Norway protocol, the voter gets weaker security guarantees: For vote privacy, it has to fully trust the server delivering the private keys (instead as before, only a coalition of two servers). For integrity, more or less the same (weak) guarantees as before result. It should additionally be noted that some keys are used multiple times within different context, which is usually not a good idea.

Scytl by Swiss Post (2016-2019) Swiss Post announced in 2015 that it would continue to provide internet voting for Neuchâtel [225], effectively taking over the further development and operation of the Scytl platform. In 2016, it was used for the first time [226], and in 2017 it was certified for 50% of the electorate (hence fulfilling individual verifiability) [227]. Post continued to invest in development, and aimed for certification of 100% of the electorate by 2019 (by fulfilling universal verifiability) [227].

The protocol is very similar to what was in use in Neuchâtel. More explicitly described is how the shuffle before the decryption is performed: Using the Bayer-Groth verifiable shuffle, which provides efficient proofs of correctness [17].[69] Further, the authentication mechanism was reworked, as the protocol is no longer tied to Neuchâtel's Guichet unique. The voter now instead downloads a key storage with its private keys, unlockable with a password received over mail [263].

While this system was in use in several cantons, Swiss Post aimed to add universal verifiability to the system. Computational and symbolic proofs of the further developed protocol were created (for both vote privacy [262, 264] and verifiability properties [94, 93]). An early version of these proofs were reviewed by Basin and Čapkun, both professors at ETH.[70] Many documents and accompanying material remain largely inaccessible.[71] The approach taken to fulfil universal verifiability seems to be to log all operations together

[68] With the key belonging to the code generator to make its involvement required when generating the finalization code.

[69] Correctness was likely a concern with the previous implementation used in Norway, as the authorities decided to let another verifiable shuffle run in parallel in 2013 [31]. Note that the Bayer-Groth verifiable shuffle was published in 2012 [17].

[70] Their report was published in May 2017, while the proof documents are dated to 2018.

[71] The proofs without supporting files are published on https://gitlab.com/swisspost-evoting/e-voting-system-2019.

with zero-knowledge proofs of correct execution (like shuffling, decryption), and then having auditors investigate these logs. Further, the RCG is divided into multiple components, as is the tallying component when shuffling the votes. We describe an evolution of this protocol in detail when we present the Swiss Post scheme of 2021.

In 2019, a public intrusion test (PIT) was organised, besides the proofs another necessary requirement to apply for the certification [35]. Issues concerning the server configuration were uncovered [72] and old dependencies were found[73], but no issue of severity higher than "LOW" was uncovered as part of the PIT [228, 285].

However, as part of the PIT, Swiss Post was also required to publish their source code [230]. To access the code, restrictive usage conditions had to be signed. The code was criticised as badly documented and hard to read [45]. Nevertheless, three critical vulnerabilities were found [44]. The first two compromised the verifiable shuffle (weak randomness generation instead verified pseudo-random) and the decryption (weak Fiat-Shamir instead of strong Fiat-Shamir) and therefore both impacted universal verifiability. The third issue affected individual verifiability and therefore also impacted the already deployed system, again through using weak instead of strong Fiat-Shamir [251, 174, 175, 285, 142].[74] As no malformed votes were found in the past elections, it is unlikely that the attack on individual verifiability was used [231, 285].

As a reaction to the found issues [59], the federal chancellery procured three external audits over operations, implementation and cryptographic protocol of the Post system. In operations, the auditors found only minor issues which remain unpublished [16]. The code review detected besides other issues big deviations between the specification and the proofs, and found that the server could violate vote secrecy. They also lamented that relevant parts of the specification are confidential, and the published portion of the code cannot be compiled and tested [179]. The audit on the cryptographic protocol revealed an incomplete individual verifiability proof, gaps in the specification which could lead to attacks, and formulations which seem to imply stronger security guarantees than actually provided. To demonstrate their claim of overboarding complexity, they propose an extremely simple voting system based on the same assumptions which reaches the same security guarantees [222].

[72] Headers were wrongly configured, including the CSP-header, X-Forwarded-For header, HSTS header, Expect-CT header, X-XSS-Protection header and the Content-Type header. Further, old TLS cipher suites were allowed. One request end-point accepted text/plain, which had the potential to allow for CORF (while no concrete issue could be shown.)

[73] An end-of-live AngularJS version was used, and an out of date bootstrap version

[74] We recommend the interested reader to focus on the reports from Haines et al. [142] and Scytl [285].

Due to the vulnerabilities, Swiss Post was not allowed to provide it as planned for the next election in May 2019 [41]. These events also contributed to the decision of the Bundesrat in June 2019 to postpone introducing internet voting as a regular voting channel [42]. In July 2019, Swiss Post announced to focus solely on providing the system with universal verifiability guarantees [229].[75]

Scytl in Australia (since 2015) New South Wales (AU) employs an internet voting system from the same vendor, primarily targeted at foreigners and voters with disabilities [143]. In general, New South Wales is intransparent about its voting processes[76], with revealing some of the details (like the source code of the used systems) constituting a criminal offence [276]. New South Wales operates the system for other states in Australia, notably for Western Australia [1].

In 2015, a substantially different protocol than the Norway/Neuchâtel approach was employed, although only a high-level specification is published (notably, verification is done by phoning the election authorities and having the plain vote read back). Teague describes severe conceptual privacy and verifiability limitations, that the implementation did not match even the high-level specification, and how the server was vulnerable to recently published and unpublished downgrade attacks during the live election [143]. The same system was reused in 2017 in Western Australia (AU), with additional problems introduced with the (essentially ineffective) usage of a DDoS prevention service [89, 106].

In 2019, the system was used again in New South Wales (NSW), now using a protocol similar to sVote of Swiss Post. It suffered from the same verifiable shuffle issues, discovered while the election was active [78]. After the vote, it was additionally revealed that the weak Fiat-Shamir issue also applied [280], although NSW asserted the contrary at the time of discovery [79]. Scytl then published a report how such an attack would have been detected (with the same arguments as used in Switzerland) [156].

Swiss Post system (since 2021) After the failed public scrutiny test in 2019, resulting in Swiss Post having to unpublish their internet voting system [41] and the Bundesrat to postpone introducing internet voting as a regular voting channel [42], Swiss Post announced to focus solely on providing an internet voting system with universal verifiability guarantees [229].

[75] For timed artifacts around the several controversies of the Swiss Post internet voting system consult https://cva.unifr.ch/content/swiss-post-e-voting-system-timeline.

[76] For a summary from the point of view of Prof. Teague see https://pursuit.unimelb.edu.au/articles/where-s-the-proof-internet-voting-is-secure.

In 2020, Scytl went in default, and while parts of it were being sold off in its bankruptcy proceedings [173], Swiss Post bought the rights of the source code. It announced to continue development in-house and to provide the new system beginning 2021 [232]. The 2021 system is therefore again an evolution of their 2019 system, respectively the Norwegian protocol. The code base also remains largely the same, although some effort to clean up has clearly been spent.[77]

We now describe this for now last evolution of the Scytl protocol. The protocol is rather long, so we only give a tight summary here. The detailed protocol is described in the appendix (see appendix B).

The voter communicates over the voting device to the voting server, which in turn handles communication between all server-side entities. Additionally, a fully trusted printer is used to prepare the election key material together with the server-side entities.[78]

First, each return code control component (CCR) chooses a secret, and agrees on a shared key-pair with the other CCRs for each voting option. The printer picks a secret key and a short secret for each voter. The printer uses the secret key to generate the pre-return codes, and the secret to generate the pre-finalization code. Then, the printer "blinds" these pre-codes, to prepare them for further processing by the CCRs. Each CCR uses its secret to homomorphically encrypt the (blinded) pre-codes. The printer multiplies the result of all CCRs and then unblinds and hashes them. Now the printer picks random (user-friendly short) return codes and a finalization code, which are associated to the hashes derived before, and encrypted under said hash. Further, the printer picks a random (user-friendly short) password, and uses it to encrypt the voter secret key into a key-store. Lastly, the printer and the mixing control components (CCMs) jointly generate an election key, with each CCM keeping a part of the secret key, and the printer's secret key part distributed to trustees.

Once the setup phase is finished, the voting phase can start. The voter receives a voting card with the its voter secret, the user-friendly short return codes and finalization code, and the password (which they use to access their voter secret key from the key store). The voter picks their favourite candidates, encrypts them under the secret key and then under the voting-option public keys, and sends this to the CCRs. Each CCR stores the vote, and partially decrypts it using their parts of the voting-option secret keys. The CCRs combine the partial decryptions to first reconstruct the pre-return codes, and then continue homomorphically encrypting it using their secret.

[77] The code base is still extremely complex, with around 160k lines of code (LoC) of Java, 30k LoC of JavaScript and more LoC for other technologies (estimated using `cloc`).

[78] The terminology "printer" is used to comply with the Swiss regulation, which in its version until 2021 does not know a setup component (but allows a fully-trusted printer).

The voting server multiplies all these results and can then derive the (user-friendly short) return codes. The voter ensures the codes match with their selections, and then enters the voter secret. The CCRs reconstruct the pre-finalization code, then again homomorphically encrypt it using their secret. The voting server again multiplies the results, and then derives the (user-friendly short) finalization code.

At the end of the voting phase, the votes are tallied. All confirmed votes are extracted, and the CCMs one after the other shuffle-decrypt the votes, with the last CCM using the secret key parts distributed among the trustees [235].

At the start of 2021, Swiss Post initiated a private bug bounty program over the system for selected researchers. In July, the federal chancellery presented the eight parties which will review the system on their behalf [48]. The general public was included in the review since September, when Swiss Post published the proofs and source code [233, 234]. Overall, the transparency efforts from Swiss Post have notably increased, and both the specification as well as the source code have undergone revisions which contribute to their improved auditability. A sign-up is no longer required to access the material, the bug bounties have increased, and the provisions no longer allow Post to keep issues private indefinitely[79]. However, the system remains very complex, and Swiss Post likely did not have the capacity to fully rework it.

Besides minor issues, researchers found an issue in the protocol which would allow to break individual verifiability (see issue #2) and an issue which broke vote privacy (see issue #11) [234].

CHVote The CHVote project was launched in 2016 by the canton Genève to replace its first generation system [113] for CHF 4.5 million [265]. The specification is developed at Bern University of Applied Sciences in their E-Voting group located in Biel [139].[80] It takes into account Swiss particularities and provisions mandated by law, for example voting cycles, variating eligibilities of voters, announcing whether a voter has participated or not, and considering one of the trustees as "administrator" to represent the cantonal administration.

The specification aims to be self-contained and fully-detailed, even providing necessary mathematical and cryptographic background information. While an implementation of the protocol is fully described, "(...) entirely missing are proper definitions of security properties and corresponding formal proofs that these properties hold in this protocol." Since the last symbolic and computational proof of the verifiability guarantees were made over

[79] All these issues were present in the 2019 review.

[80] The group's webpage can be found at https://e-voting.bfh.ch/.

version v1.3 [24], the specification has seen substantial updates. We describe here version v3.2, published on 14.12.2020 [139].

The trustees first generate an threshold public/private key pair for the election, with each trustee having its own share of the private key and the full public key. Further, for each voter, two additional threshold public/private key pairs are generated. Additionally, each trustee generates for each voter a polynomial of degree $k-1$ for k the number of candidates the voter can vote for. Then, the polynomials are each evaluated at n points for n the total number of available candidates. The trustees finally send the private key shares of the voter private keys and the n points to the printing authority.

The printing authority now constructs a voting card for each voter: The private key shares and the n points of each trustee are combined and result in two credentials and n return codes for each voter. Further, out of all return codes, the finalization code is generated. The printing authority now prints the voting card and sends it over postal mail (assumed to be a secure channel) to the voter.

The voter selects its candidates, then creates an encryption of its selection with the election public key. The voter sends it, together with the first of their credentials, to the trustees. They validate the credential and store the vote. The vote also serves a double-purpose as a k out of n OT query: Each trustee answers this query with k out of n points of the voter-specific polynomial. This core idea of the protocol, namely to use oblivious transfer for cast-as-intended verification, was also published independently and peer-reviewed [138].

The voter combines these OT query responses into the return codes (in the same way as the printing authority did), and ensures the expected return codes match. The voter then reconstructs the polynomials out of the OT query responses.[81] The polynomials are each evaluated at 0, and this result is sent, together with the second credential, back to the trustees.

The trustees confirm the polynomial evaluation matches their expectation, and reply with the rest of the n points of each of the voter-specific polynomials. The voter can combine these (same as the printing authority did at the start of the protocol) into the finalization code. If this succeeds, then the voter knows everything was successful, and no further actions from their side is necessary. If the constructed finalization code does not match the finalization code printed on the voting card, then the voter is instructed to trigger an investigation. If any previous step of the voting procedure fails, voters are instructed to abort and use a different voting channel.

[81] Note that this step only succeeds when k *different* candidates were selected, else the voter cannot reconstruct the $k-1$ degree polynomial. This step hence confirms the submitted vote was properly constructed.

Finally, after the election, the trustees shuffle and decrypt the votes. While announcing the counted votes, the trustees also publish finalization codes of those voter that did participate, and abstention codes for those voters that did not.[82] Any voter may now check they are correctly tracked as voters or non-voters [139].

CHVote features individual verifiability, universal verifiability and ballot privacy. Critical processes are jointly executed by trustees, which cannot fake results, but any trustee may prevent further execution of the protocol by refusing to participate. The single points of failure is the printing authority (which can both impersonate voters, as well as break their privacy), and the voting client (which can break voter privacy).

Up until at least v1.4.1, the canton Genève funded an implementation meant to the be used in production [119], before the project was stopped due to financial reasons in 2018 [114, 115]. Genève then proposed that the federal government should continue developing and operating the system, which was denied by both federal chambers [129]. The implementation [119] has not received much attention, likely as there is no clear path how it is ever going to be used. Superficially, it looks high quality and properly documented.[83] Genève also published parts of the source code of the previous generation system, which was actively used in elections, but had weaker verifiability guarantees [112].

The E-Voting group continues to develop the CHVote specification and publishes an up-to-date reference implementation under the name OpenCHVote [108]. The OpenCHVote implementation focuses solely on the protocol aspects, hence for example no user interfaces are provided [108]. Porting the application to be production ready would require significant work.

3.3 Mechanisms

We review mechanisms we encountered in our literature review. While we aim to give a broad overview of what mechanisms have been found useful, we do not argue for completeness.

The mechanism usually have to balance between guaranteeing correctness and confidentiality, or in electronic voting terminology, verifiability and privacy. Depending on the mechanism chosen, the necessity of trust assumptions or impossibility of security property automatically follow.

[82] Abstention codes were also generated in a distributed manner (essentially XOR-ed random values of all trustees), and printed on the voting card.

[83] However, also this code base is very complex, with around 70k lines of code (LoC) of Java, 40k LoC of JavaScript and more LoC for other technologies (estimated using `cloc`).

3.3.1 Authentication mechanisms

We discuss mechanisms with which the voter can authenticate themselves. For verifiability, authentication must not be easily forged. For privacy, the authentication should not reveal the identity of the authenticated voter.

We name the authority which stores the cast and authenticated votes the collection authority. Depending on the mechanism, we may also employ an authentication authority which is able to authenticate individual voters.

Signatures The voter may simply sign the vote and then send it to the collection authority (this is what is used in Estonia [148]). However, the signature necessarily binds the vote to some public key, and if this public key is attributable to the voter, then so is the vote.

To avoid attribution, the signature could instead stem from some (trusted) authentication authority. The voter sends the vote to this authentication authority and authenticates, and the authentication authority replies with a signature over the vote. The voter then sends this voter-independent signature together with the vote to the collection authority.

To relax the trust assumptions on the authentication authority (which in the scenario described above learns the vote of the voter), we may employ a blind signature scheme. The voter only sends their "blinded" vote (hence with its content hidden) to the authentication authority and receives back a signature over it. The voter then unblinds the vote (hence reveals its content) and sends it together with the - still valid, as guaranteed by the blind signature scheme - signature to the collection authority. This mechanism was originally proposed for the FOO scheme, one of the first voting schemes ever [124].

Token-based authentication Instead of signatures, the voter may also submit some form of authentication token. If the token is not bound to some identifiable voter, this easily preserves privacy.

If the token is not bound to some specific vote, then the voter has to be convinced by some other mechanism that the appropriate vote was perceived as authenticated by the collection authority (see the verification mechanisms in section 3.3.3), else circumventing the authentication by submitting a different vote is likely trivial.

3.3.2 Casting mechanisms

We discuss mechanisms with which the voter can cast their vote. An optimal casting mechanism enables the voter to not reveal their preference to anyone but the tally. We might however need to compromise out of usability

concerns or because other mechanisms used in the voting protocol require the cast vote in a specific format.

In this subsection, we only care about privacy properties. Whether the cast vote has been received at the collection authority as intended is taken care of by the verification mechanisms (see section 3.3.3).

We assume the voter uses a device to send the vote to the collection authority. We discuss the mechanisms depending on whether the voter's device is trusted or not.

With trusted device When the voter has access to a trusted device, they can enter the vote in plain. This might be an advantage from a usability perspective, and further enables more complex transformations of the vote until it is cast to the collection authority (e.g. encryptions). However, equipping each voter with a trusted device might be prohibitively expensive, and the trust assumption might be too strong.

Without trusted device Even without having access to a trusted device, the voter may still be able to cast their vote while preserving privacy.

The voter might transform the plain vote into something which is not interpretable by the adversary. In code voting schemes (like SureVote [68]), the voter maps their plain vote to a voting code, with the association unknown to the adversary. Voters might even be able to perform simple mathematical operations like XOR or clock addition which enable perfect encryption.[84]

Besides encrypting the vote, another approach hides the context in which the vote was cast. In Scantegrity, it is publicly published which boxes on the ballot sheet were checked, but it remains hidden how the paper ballot looked (hence which box belongs to which candidate) [64].

Another idea uses indistinguishability of values picked by the voting authority and the voter: The voting authority assigns each candidate a value, then sends this dictionary over a secure channel to the voter. The voter replaces the value of their favourite candidate with a different value, then submits the whole dictionary to the collection authority. As long as the adversary cannot tell who picked which values, it will not learn the true vote. This mechanism was proposed as Bingo Voting [27].

3.3.3 Verification mechanisms

We discuss mechanisms with which the voter can verify their vote was cast and confirmed as intended. While the voter should be convinced the verifi-

[84] Possibly, using visual guides as proposed in [196].

cation indeed guarantees the vote was cast correctly, ideally the voter cannot convince third parties of this fact to guard their privacy.[85]

We assume the voter uses a device to send the vote to the collection authority.

Verified cast When it can be ensured that the vote is cast exactly as intended, then an explicit verification mechanism is not required. However, this obviously requires strong trust in all involved parties, including in the device used for casting the vote and the collection authority.

The Benaloh challenge (also called audit-or-cast mechanism) relaxes the trust in the device used for casting. It works as follows: The device encrypts the vote, and then commits to the result. The voter can then decide whether they want to cast the vote or choose to audit the vote. When choosing the latter, the device uncovers proof that the computation was performed correctly (for example, the randomness used in the encryption).[86]

Receipts The voter receives a receipt for the vote they have submitted. The receipt might be the same for all votes (confirming only that some vote was registered), or might be tied to a specific vote (for the stronger proof that the specific intended vote was registered). In its trivial form, like a signature over the received vote, or some pre-agreed token, this would make it easy for the voter to also proof the fact to the adversary. There are many ideas how to counter this privacy issue.

Forgeable receipts allow the voter to make a receipt look like it belongs to a different vote (like Selene [255]). Masked receipts or votes, of which either has been (re-)encrypted, cannot be decrypted by the voter, but their validity can still be checked by auditors (like in BeleniosRF [66]). Probabilistic receipts may only guarantee parts of the vote, but as the collection authority does not know which part was handed over as the receipt, it must still store the whole vote as expected (like Three Ballots [247] or Scantegrity [70]). Floating receipts are receipts which are passed on to different voters, hence voters ensure that the vote of some other voter is correctly stored by the collection authority (like Floating Receipts [247]).[87]

Constructing the receipts is also a good opportunity to ensure multiple authorities (of which possibly only a single one is trusted) have indeed seen the same correct vote. For example, the receipts possibly need to be thresh-

[85] This is problematic as voters might be incentivized to sell their vote.

[86] Note that to verify the proof, the voter requires as second non-colluding device.

[87] This still prevents vote selling as the voter cannot prove their own vote, unless the receipt is "floated" to the adversary, against which the scheme has to defend appropriately.

old decrypted (like in Pretty Good Democracy [258])[88] or jointly constructed (like in industry schemes [139, 235]).

Another approach to make the receipts useless to the adversary is to allow fake votes. However, this has to be done in a way that the voter is still convinced what their final submission is, which might be difficult to understand for voters [198].

Another possibility is vote updating, hence the voter is able to change their vote by casting another one (see [284] for a discussion). This may also be enabled over conventional channels like postal voting or supervised voting. In that case, the privacy of vote updating is guaranteed through the intransparency of the conventional channel, although the problem persists how the votes of the electronic voting channel are filtered out in a verifiable way without the attacker noticing [137].

3.3.4 Tallying mechanisms

We summarize mechanisms used for tallying. The mechanism needs to ensure the tally is executed correctly. At the same time, the tally often also has to establish or preserve privacy properties: Votes may still be attributed to individual voters, hence a direct decryption of votes would break ballot privacy. Further, votes must not be decrypted before the voting phase finishes, as this would break fairness.

Threshold and verifiable decryption To avoid early decryption of votes one may employ a threshold decryption procedure. Instead of a single decryption key, shares of the decryption key are distributed to multiple authorities, which are all required to decrypt the ciphertext. There exist even more general k-out-of-n decryption schemes which ensure malicious authorities cannot prevent decryption as only k out of n key shares are required.[89]

Some decryption schemes also support verifiable decryption: Besides the result of the (partial) decryption, a proof is produced which guarantees the decryption has been executed correctly. This proof might even be zero knowledge, hence avoids revealing the used decryption key, or anything more about the processed ciphertexts which was not known beforehand.

Privacy-preserving tally Depending on the voting scheme, individual voters are still connected to their votes. This link must be separated before decryption or ballot privacy would be broken.

[88] Bonus points for Pretty Good Democracy: The trustees decrypting the receipts do not learn what the cast vote represents, only that it represents a valid voting option.

[89] For example using Shamir's Secret Sharing.

Shuffles take a list of votes as input, and output them in a random order. To avoid the adversary being able to trivially match input and output ciphertexts, the votes need to be reencrypted. If the authority performing the shuffle cannot be trusted, verifiable shuffles produce proofs which guarantee the output ciphertext represents the same plaintext as the input. Then, the shuffle is executed by multiple authorities one after the other, each operating on the output of the previous shuffle. If at least one of the authorities is honest, then the link between voter and vote is separated. Shuffles of this kind are for example employed by the schemes used by industry [139, 235].

Instead of shuffling, one may also encrypt the votes using a homomorphic encryption scheme, then calculate the tally result directly over the ciphertexts. Only the final ciphertext is decrypted (representing the tally result) and therefore no link to individual votes can be established. Homomorphic tally has for example been proposed for Prêt à Voter [253].

Verifiable tally When one wants to ensure the tally is executed correctly, but is able to trust the tally authority to preserve privacy, additional mechanisms become possible. In general, the tally authority is asked to tally, to commit to partial results and only reveal the final result. Then, to ensure the final result is indeed correct, the tally authority is asked to uncover (in a manner unpredictable to the tallying authority) some of the partial results. If constructed cleverly, besides proving correctness of the tally result publicly, ballot privacy of voters remains preserved.

One mechanism allows the voters to submit multiple votes, which only together reflect the true intention of the voter. The tallying authority is then forced to reveal randomly some - but not all - parts of the vote and prove these parts were included correctly in the tallying result. This is similar to the split-value representation by Rabin [239, 190] which has been developed into a voting scheme [238]. Similar ideas are used in KTV-Helios [167] and in a protocol proposed by Locher et al. [178].

Another mechanism specifies the tally authority to commit to three lists, which each have the same number of entries: A list L_{auth} with the original authenticated encrypted votes, a list L_{renc} of re-encrypted votes (but without the identifiable authentication), and a list L_{dec} of decrypted votes. After the commit, the tallying authority is asked (in an unpredictable fashion) to show for each entry in L_{renc} that there exists either a corresponding entry in L_{auth} or in L_{dec}. This idea was proposed originally in [21].

3.4 Summary

We have seen the various privacy and verifiability properties demanded from an electronic voting system. Privacy ensures the vote stays private,

depending on the level of privacy even if the voter colludes with the adversary. Verifiability ensures the unmodified vote is indeed included in the final result. Properties however often lack a universally agreed to definition.

Many different proposals for electronic voting systems exist. On-premise schemes manage to reach very strong guarantees using simple mechanisms, which however are not transferable to the remote setting. Remote schemes from research are still able to provide strong guarantees, however the involved cryptography leads to high complexity, and often also low usability. Industry proposed their own schemes, focusing on verifiability and practicality, although under weaker security guarantees and again involving complex cryptography.

We further have seen various mechanism which enable a secure electronic voting system. For authentication, signatures or tokens can be used, which both have to be bound to the voter and the vote, but ideally without revealing this link publicly. Casting in a privacy-preserving way is possible even without a trusted device, when the voter can be assumed to perform simple tasks. Verification almost always requires receipts, which are difficult to construct in a privacy-preserving way. Verifiable tallying uses threshold and verifiable decryption or revealing of partial results, and for additional privacy guarantees, also verifiable shuffles or homomorphic encryption.

Chapter 4

Internet voting in Switzerland

Introducing an internet voting is a long-running project with high initial hurdles and insecurities, while it is closely observed as it impacts one of the most important processes in a democracy. Both makes the project highly vulnerable to reoccurring political debates, which might change policies towards internet voting abruptly. In Estonia, some parties were reluctant in introducing internet voting, as it might shift public participation (and therefore affect the voting results) [282].[1] The newly elected government of Norway in 2013 stopped internet voting as some of its first acts in power [31]. France abandoned internet voting in 2016 due to national security concerns [243]. In the canton of Basel in 2018, the parliament suddenly shifted against internet voting [261], then shifted back to supporting it only three months later [260].

Further, implementation is tricky. Due to the complexity of the involved systems, this takes time, is risky and requires significant investment (which provokes another heated discussion of cost/benefit of the system as a whole). In Germany in 2009, the federal constitutional court ruled against electronic voting as the inner workings were not comprehensible without domain expertise, therefore arguably violating the principle of a public vote [242]. In Norway in 2011, the system was operated by two different departments to account for its non-collusion assumption of two servers [273]. In Sweden in 2019, the internet voting project was cancelled the evening before the release date due to delays with testing the system [104]. In Switzerland in 2019, the previously used (and certified) system was taken offline, as serious security vulnerabilities were discovered in public review [41]. Internationally, so far only the Spanish firm Scytl was able to win tenders, although it faced repeated criticism over the quality of its solutions (e.g. [217, 179]).

[1] This aspect very likely affected the discussion in other countries as well, see for example the first report to vote électronique of the Federal Chancellery [54].

We analyse in this chapter the history, law and politics of internet voting in Switzerland. We call it in this context vote électronique, how it is usually referred to in the French part of Switzerland.[2] While focusing on the Swiss perspective, we give international context where appropriate.

Political debates about pros and cons of internet voting In general, there are many unknowns about the benefits, risks and cost of such a system, making a decision hard to take. Besides the arguments supported by concrete numbers (such as cost), influencing arguments may also be of a more strategic nature. In Switzerland, it is argued that the currently most used postal channel will lose trustworthiness over time as communication in general shifts online [56]. In Estonia, providing an advanced electronic infrastructure to its citizen may also be a matter of national pride.[3]

The strongest arguments for internet voting usually revolve around the inclusion of more voters. The target is either to generally increase participation as in Estonia [184], or to support voters not able to use the traditional channels (like expats, or voters with disabilities), which was the primary motivation in both Switzerland and Australia [56, 219]. However, most studies find participation to not change, rather that existing voters switch voting channels [281, 131].

The counter-arguments to internet voting system usually stem out of a perceived lack of security. In an area of active research with researchers being at odds even over the definitions of the security properties, it is hard to tell whether a system is appropriately secured or not. Some international experts even fundamentally oppose internet voting, and may stir the debate by publishing exaggerated claims over found or perceived issues. National security considerations, or the usual noise of self-proclaimed experts arguing in either direction, may also influence the perceived security of a given system. However, so far no attacks on internet voting have been reported, albeit this "result" is obviously achieved with a small sample size, and might simply be attributed to luck or secrecy.

Further hard to tell is the cost of such a system, also compared against other voting channels. In Switzerland, for the various small-scale trials, the cost was higher [56]. In Estonia, its large-scale internet voting is the cheapest voting channel per vote [163].

So far only Estonia was able to set a stable political basis for internet voting, and as a result to provide it uninterrupted at the national level since 2005 [183, 282]. Switzerland has seen multiple systems come and go [56, 58, 41, 115], and three major revisions of its applicable law [53, 34, 61].

[2]Note that electronic voting, how the term is understood internationally, includes not only internet voting but all forms of electronic support during voting [209].

[3]For example, see how internet voting is described in [282].

Implement and operate an internet voting system A suitable internet voting system does not exist of-the-shelf, but has to be adapted to the specific organisational realities and the governing law of the country it is supposed to be used in. Designing cryptographic protocols is notoriously difficult, and even harder when the resulting protocol must satisfy seemingly contradictory properties such as verifiability and privacy.

The international research community in general seems not particularly helpful, proposing largely schemes unfit for big electorates[4], or overly complex schemes archiving some hardly motivated theoretical target for security or privacy [20]. We know of no researcher-lead initiative which survived more than two voting periods nor included an electorate of notable size.

The industry reacts with proposing their own protocols, with mechanisms and tradeoffs previously not considered in academia (e.g. [6, 146]). In Estonia, a very simple scheme enabled by their national ID infrastructure was put in use, essentially signing the encrypted vote [184]. In other countries, Scytl employed its protocol based on return codes: These would be returned by the server for the specific selections a voter took, and the voter would compare them to codes printed on a ballot sheet to attest the server correct operation [7]. Both systems were gradually extended over time to meet stronger and stronger security guarantees, while the basic concept remained largely the same.[5]

4.1 History

Switzerland already has an extensive history with vote électronique, having run many trials on a variety of different systems and legal basis. We give an overview primarily based on the reports of the Federal Chancellery.[6]

Pilots since 2002 The Swiss Federal Council published the first report about how to introduce vote électronique as a major voting channel in as early as 2002. It analysed potential political impacts, such as expected shifts in the political landscape due to changes in who votes. It included a security analysis declaring the target to reach similar security as existing channels, identified several security challenges and defined technical minimal requirements to guarantee security. Further, financial impacts were analysed: It was estimated that deploying vote électronique by the municipalities is about 20x as expensive as deploying it by the federal government or the cantons [54].

[4]Due to performance constraints (e.g. [75]) or inherent usability issues (e.g. [2, 75, 255]).

[5]For Estonia, see [184, 150, 148, 145]. For Scytl, see [7, 126, 263].

[6]Timeline of the federal government: https://www.bk.admin.ch/bk/de/home/politische-rechte/e-voting/chronik.html. Timeline of Zürich: https://www.zh.ch/de/politik-staat/wahlen-abstimmungen/wie-stimme-ich-ab/e-voting.html

The canton of Genève had the first legally binding vote électronique on the 19th January 2003. Other pilot projects, financially primarily covered by the federal government, were active in Neuchâtel and Zürich. Other cantons updated the legal framework to allow vote électronique and started to centralize their voter registers (which was seen as necessary precondition for small municipalities unable to operate their own servers). While a much higher voter turnout was not expected, vote électronique for foreign Swiss Voters [7] would improve their voting conditions [8] [32].

Growth of trials 2006-2013 By 2006, the three pilot systems were fully developed, and had all seen successful, legally binding votes. Neuchâtel integrated the vote électronique system into a digital counter with additional capabilities (like tax matters or searching for license plates). Genève developed a stand-alone vote électronique system using its central voter register. Zürich's system worked with decentralized voter registers, and besides internet also offered voting over SMS. No big change in voter turnout was observed. In Genève, around 22 percent of eligible online voters decided to vote online [55].

The next report by the federal government was released in 2013. It described that the system of Zürich was now used by six more cantons (together forming the "Consortium"), and the system of Genève by three more cantons. Prioritised recipients of the vote électronique systems were foreign Swiss voters and people with disabilities. While it was recognised that the transparency of the systems should be improved, the trial runs were considered a success, and were to be extended [56]. However, both Genève as well as Zürich had realized the necessary next step was a fundamental shift of the security concept towards verifiability [56, 274].

New legislation and push towards verifiability 2013-2019 At the end of 2013, the legislation was updated: The fixed limit of 10 percent of the electorate able to participate over vote électronique was replaced by limits of 30 percent, 50 percent and 100 percent, with each higher limit enforcing stronger conditions [53]. These conditions were worked out by the Federal Chancellery, which in its ordinance required individual verifiability for over 30 percent and additionally universal verifiability for over 50 percent of voter participation [35]. Cantons, parties and other organisations welcomed the revision and largely agreed over its content [34, 33].

In 2015, the Federal Council did no longer allow the "Consortium" system to be used in the national election [290]: It had an unresolved issue jeopardizing vote secrecy and it did not conform to the new legislation

[7] 1% to 5% of the voters depending on the canton.

[8] Sending voter documentation abroad was expensive, and often arrived too late.

[58]. Shortly after, the "Consortium" was dissolved [289]. Genève continued to use its system approved for usage below 30 percent of the electorate [117]. Neuchâtel used its equally approved system for the last time, and announced its partnership with the Swiss Post for future elections [120]. Both Genève and Neuchâtel had updated their systems before the election, and now claimed their voting systems to be "second generation" (hence providing individual verifiability) [116, 120].

Genève wanted to further improve verifiability and transparency, specifically aimed to also provide universal verifiability. In 2016, funding of around CHF 5 million was allocated to develop the next generation of its vote électronique system [265]. To develop the specification, Genève then entered a collaboration with the Bern University of Applied Sciences. Their E-Voting group published the version v1.0 of the specification, called CHVote, in 2017. It featured individual and universal verifiability and distribution of trust between the server components (the only remaining fully trusted single component was the printing authority) [139]. The involved researchers notably developed UniVote already in 2013, which was an individual and universal verifiable voting system used by student associations in Bern and Zürich among others [102]. The canton Genève started to develop the new system and published the source of a proof of concept on GitHub [118]. However, at the end of 2018, Genève again stopped the development due to financial reasons [114]. The code developed until then was published on GitLab [119].

New legislation and everything stops 2019 In 2017, the Federal Council decided that the time is ready to introduce vote électronique as an ordinal voting channel [37]. Following a report by an expert group deeming Switzerland ready for this step [38], a consultation round starts at the end of 2018. It aimed to introduce the legal foundations to transfer the trial vote électronique into regular service. The limits on participation were to be removed, but only both individual as well as universal verifiable systems were to be deployed [39, 40].

During the consultation round in spring 2019, a public review was performed on the Swiss Post internet voting system. The code was criticised as badly documented and hard to read [45], and three critical vulnerabilities were found [44]. One of the issues impacted the currently deployed system, and Swiss Post was therefore not allowed to provide it as planned for the next election in May 2019 [41].[9] The results likely came as a surprise to pol-

[9]For timed artifacts around the several controversies of the Swiss Post internet voting system consult https://cva.unifr.ch/content/swiss-post-e-voting-system-timeline.

itics[10], administrations[11] and Swiss Post, although systems of the supplier Scytl had experienced serious issues before (e.g. [217, 213]).

When the consultation round finished in June 2019, the political landscape had changed drastically compared to the last consultation in 2013. Considerable doubts had risen: While most cantons (20) still supported the new legislation, all political parties and some major organisations were opposed. They felt the time for the decision is not ready and criticised the lack of transparency and understanding of the current systems (including for non-experts). Some also doubted the cost/benefit of vote électronique in general, or wanted that the state itself provides such a system rather than private companies [46]. End of June 2019, the Federal Council decided to abort the introduction of the new legislation, and to instead rework the vote électronique concept [43].

Expert dialogue and new legislation 2020-2021 By the end of 2019, no vote électronique remained available to the cantons. After Post stopped providing its individual verifiable system, Genève informed in June 2019 that it will also no longer provide its already deployed system, effective immediately [115].

Two initiatives remained aiming to bring vote électronique back: The Swiss Post invested in improvements over its universally verified system [232], and the Bern University of Applied Sciences E-Voting group continued to develop the CHVote specification, providing a partial implementation under the name of OpenCHVote [136, 108].

In the mean time, the Federal Chancellery organized a dialogue with experts from industry and science. They motivated improvements with security, namely to use standard cryptographic primitives, a diverse set of implementations and to consider using a public bulletin board. Also more transparency (for example due to higher quality source code) and independent reviews were suggested. Based on this feedback, and the experiences of the past years, a long list of concrete improvement measures was proposed and compiled into a report [47]. Using this report, the Federal Council decided to further pursue vote électronique based on new legislation to be made in 2021 [59].

In April 2021, the consultation procedure started aiming to reform the VPR and VEleS to further strengthen the security requirements [61, 52, 50]. It closed in August 2021, with most cantons and political parties as well as

[10] The Federal Council reacted by procuring three additional external audits over the Swiss Post system, which confirmed the found issues [16, 179, 222].

[11] Administrations using the Swiss Post system had no contractural agreements over what happens if the system fails public review [144].

many organisations submitting their opinions. The participants largely supported pursuing the strategic goals (continued development of the systems, improved audit and transparency, stronger link to research) and agreed that the measures taken will achieve these goals [51].

In 2021, Swiss Post started a bug bounty program to let researchers review its improved system. Additionally, the Federal Chancellery presented eight parties which will review the system on their behalf [48]. So far, at least two critical vulnerabilities in the protocol were uncovered [234].

4.2 Legal

While cantons organize votings, usage of vote électronique channels requires approval of the Federal Council (Art. 8a of BPR [98]). The Federal Council defines in its ordinance about political participation VPR some basic requirements for vote électronique (Art. 27 of VPR [53]), while much of the competence for the more technical implementation of the law is transferred to the Federal Chancellery. Votes must stay secret and each voter can only vote once. Different level of approvals exist, while with increasing size of the electorate also the security requirements rise.[12] The voting results have to be checked for plausibility. Voters must be able to understand the verifiability concept, and must know how to react when problems arise. Voting systems may also be provided by private firms. If the overall concept of the cantons is convincing, the Federal Council grants its basic approval. Then, the cantons apply for a specific approval for each election at the Federal Chancellery.

The Federal Chancellery formulates out the technical requirements on which approvals are granted in its own ordinance VEleS [35]. Any system requires foremost secure and trustworthy voting, ease of use and comprehensive specification (Art. 2). Depending on the size of the electorate allowed to participate over the online voting channel, additional constraints are enforced: For over 30 percent of the electorate, individual verifiability is required (Art 4), and for over 50 percent, additionally universal verifiability must be enforced (Art 5).[13] It dictates risk analysis and risk minimization, an external audit of the system and publication of the source code. Further technical details are documented in the comprehensive appendix, which notably includes the requirement of both a cryptographic and symbolic proof of the used protocol.

[12] This is the central concept changed in the newest revision of 2021, which requires the same (high) security requirements independent of the size of the electorate [50].

[13] The newest revision of 2021 requires both individual as well as universal verifiability independent of the size of the electorate [50].

Cantons need their own legal basis for vote électronique in their constitutions, laws and ordinances. Most cantons have created that basis until now [57].[14] About half have experienced at least some form of vote électronique[15] [214, 56].

Reform of VPR and VEleS started in 2021 The new VPR and VEleS further strengthen the security requirements. During the consultation round in summer 2021, the strategic goals to achieve this were named to be continued development of the existing systems, more effective control, improved transparency and tightened the connection to research. As it is customary, a report was provided explaining the purpose of the changes [50].

The VPR now requires all systems to support both individual and universal verifiability, independent of the size of the electorate. The electorate must never surpass 30 percent of cantonal voters, respectively 10 percent of the national voters. The transparency and review requirements are further strengthened, too. Notably, private providers continue to be allowed to provide systems [61]. The VEleS is also reformed: It is now much more detailed, allowing less freedom when designing the protocol[16], writing the source code, operating the infrastructure and publishing the specification. Notably, an open source license requirement is still missing [52].

The new provisions in the VEleS primarily try to enforce a "high-quality" system, presumably as a response to the arguably low-quality system observed during the Swiss Post review of 2019.[17] It is unclear whether these new provisions are able to achieve the desired outcome, as defining "high-quality" in legal terms is very challenging. Not surprisingly, some of the formulations are impossible, way too strong or not strong enough.[18] The final revision is supposed to be published in the summer 2022 [60].

4.2.1 International institutions

International law and organisations also consider vote électronique, which then impacted the Swiss law and processes. We present the most relevant international institutions with an application to vote électronique.

[14] By 2015, still six cantons have not: AI, AO, NW, TC and ZG [214].

[15] AG, FR, GL, GR, SH, SO, SG, TG, ZH with the Consortium system; BS, LU, BE, GE with the Genève system; NE with the Neuchâtel system [214].

[16] For example, it explicitly requires that for individual verification a proof is displayed to the voter, which is checked by the voter against previously received printed out codes.

[17] In 2019, concrete issues breaking individual and universal verifiability were uncovered. If none of these would have been found, the administration likely would have had no legal basis to deny usage of the system, even though the public review seemed to indicate the low quality of the reviewed system.

[18] For example, it is required to unit test "all valid inputs" (see Art. 25.13.2; impossible), the source code contains "no repetition" (see Art. 25.11.4; too strong), but version control information is not required to be published (not strong enough).

Universal Declaration of Human Rights (UDHR) The UDHR by the UN states that "(...) [elections] shall be by universal and equal suffrage and shall be held by secret vote or by equivalent free voting procedures." [11]. Universal suffrage means that any voter is able to vote, while equal suffrage ensures each voter has the same amount of votes. Secret voting requires the vote to not be known to others except the voter, and free voting ensures no votes are bought, sold or otherwise traded.

The UDHR is very abstract law, which then motivates more specialized laws. In Europe, the Council of Europe is the main reference which applied the UDHR to vote électronique.

Council of Europe E-Voting Recommendations The Council of Europe[19] already published its first recommendations on e-voting in 2004 [207]. The recommendations are based on the UDHR and similar international law, and aim to guide states implementing e-voting[20]. After around a decade in effect, an update was recommended to adapt to progress made in practice and research (for example, verifiability is not part of the 2004 regulation) [100]. 2017 the second version was published, which restructures and overhauls most provisions [209]. It is argued by the authors that the recommendation will develop binding character, when courts start to seek it out to resolve controversial national election matters [189].

The recommendation is structured by universal, secret, equal and free suffrage, motivated by the applicable paragraph in the declaration of human rights. Further sections deal with regulatory requirements, transparency measures and reliability of the system. For each section, multiple abstract sub-targets are spelled out [209]. These are expected to stay the same for the foreseeable future [189], and in fact contain only few e-voting specific provisions.

To put these abstract sub-targets into practice, a guideline is provided which details for each sub-target how to implement it in more concrete and technical terms. It may also name other concrete standards to fulfill, for example to comply with the guidelines set out by the Web Accessibility Initiative (WAI), for the sub-target of ensuring the persons with disabilities are able to use the voting system, in the chapter of universal suffrage. It also specifically lists individual and universal verifiability as a means to implement free suffrage, or receipt-freeness as means to achieve secret suffrage. Further, best practices for processes are proposed, like physically destroying data drives which contain to-be-deleted data. Additional provisions require to enforce transparency, effective certification procedures and precautions to

[19]Note that the Council of Europe focuses on human rights, compared to the primarily economic focused European Union, and the primarily security focused OSCE.

[20]The focus is not exclusively on internet voting, but on all forms of electronic voting.

ensure system availability [208]. The guidelines will be regularly updated to reflect current best practices [189].

Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights (OSCE/ODIHR) The ODIHR, upon invitation, creates assessments of election procedures.

Switzerland requested one of these assessments in 2007 for its federal elections for the national council and the council of states. Interesting in the context of this work is the description of how postal votes are validated: It is only ensured the registration card exists and is signed. No further checks are performed, specifically "The signature is not compared to an existing control signature, nor is the electoral register marked to show from whom a ballot has been received.". For in-person voting, no additional checks are performed. Further noted in the report is also how the deadline of postal voting material to be arrived at the voter is only ten days, leaving little reaction time, especially for expats, to notice a missing delivery [211].

In 2011, another assessment was requested, with overall very similar results as the previous. An increased effort to ensure expats receive their voting material on time is described (due to problems with late-deliveries in 2007). Small issues with lacking vote secrecy in polling stations were also discovered [212].

For the first time, the report also analyses the vote électronique procedures. It noted that the law is not very precise with its security requirements and when assigning responsibility, especially if one canton hosts the system, and another one uses it. It criticised (too) minimal oversight when printing the voting cards (which contain confidential information), and proposed to use seals in the Genève system to strengthen voter protection. It recommended to decrypt the votes only on election Sunday (and not one day before, as it was practice). It lamented the lack of verification for the voters whether their vote was registered correctly, and generally recommended to adopt systems which are more on-par with research proposals. When managing the systems, the report mentioned that the cantons were entirely dependent on external experts, and recommended to build up internal competencies. It further noticed that in the Consortium system, the election key was stored by the external operator, and instead recommended it to be distributed to multiple likely non-colluding parties. It regretted that no end-to-end tests are performed pre-election, and certification procedures miss entirely, contradicting federal law. Additionally, the performed audits were not public, and for the Consortium system, not done over the latest update of 2010. A general intransparency about the systems and the administrative processes was lamented [212]. 22k expats[21] were allowed to use the vote électronique

[21] From AG, BS, GR and SG.

(while voters living in Switzerland were not yet included) [214].

For the next national elections of 2015, the report focused entirely on vote électronique. It recognised the law and processes were reworked, and most recommendations from 2011 were implemented. The introduction and approval processes for the systems were largely positively described. To improve transparency, the report recommended to let voters try out the new systems apart from a real election, and to publish the results of audits.[22] It further recommended easier access to the source code and the use of open standards.[23] The OSCE recommended again, as in previous reports, to allow vote overriding for improved voter coercion mitigation. 132k voters were allowed to use the vote électronique, of which 34k were expats.[24] Per canton, around 15% then did actually use the vote électronique channel [214]

In 2019, a preliminary report was created before the election. The biggest critique, as in the two reports before, was the intransparent party financing [216]. The recent developments in vote électronique are summarized, and no additional issues were identified. Although the preliminary report finished with a recommendation to observe the election, no such mission was sent due to limited resources on ODIHR's part [275].

4.3 Political

Around the year 2000, political interest in vote électronique was high [54]. The federal government established the legal foundation and provided financial support, which enabled the introduction of three different cantonal systems in only a few years [32] and many trials by 2006 [55]. Then, the political interest slowed down, presumably as there was no change in voting behaviour, and no problems arouse using this new channel. By 2012, political interest in the topic was perceived as slim [212]. In 2013, the limits of the electorate able to participate in electronic voting was increased, to favour systems that invested in individual and universal verifiability [35].

In 2015, the Consortium system (the only one out of the three systems not investing into verifiability [116, 120]) was not granted permission to be used any more, and the number of cantons being able to provide vote électronique suddenly dropped [290]. This seemed however not to impact the overall progress, as already in 2017 the Federal Council decided the time is ready to fully roll out vote électronique [37]. The cantons supported this progress in general, although some doubts were raised concerning the security and cost-effectiveness of the new voting channel [240].

[22] Only the Genève system published the audit results, while Neuchâtel did not.

[23] To improve interoperability with other election software. BE discontinued the Genève system as it was incompatible with its tabulation software [214].

[24] From BS, LU only expats; from GE, NE including voters living in Switzerland.

However, during the consultation round in spring 2019 to establish vote électronique as an ordinal voting channel [39, 40], politics suddenly took a strong interest. First, Genève decided it will stop providing its system due to its cost [114], leaving Swiss Post as the only provider. Shortly after, Swiss Post underwent a public review which uncovered critical vulnerabilities [44] and the system had to be withdrawn from the next elections [41]. Considerable doubts about whether Switzerland is ready for this new voting channel arose: While most cantons still supported the roll out, some major political parties and civil societies were now opposed [46]. The Federal Council aborted the revision, and announced instead to rework the vote électronique concept [43].

After a dialogue with experts [47], another revision started in 2021 to further strengthen the security requirements and transparency measures [61, 52, 50]. Most cantons and parties participated, and now largely agreed over the measures taken, which notably again included a strong restriction of electorate able to participate [51, 49]. The new legislation will be published mid 2022 [60]. A public initiative to stop vote électronique altogether for at least five years did not materialize.[25]

In political debates, vote électronique is primarily compared to postal voting. For example, when discussing security, similar guarantees than postal vote are politically requested.[26] This mindset, and the pro- and contra arguments arising out of it, seem to be almost the same now, as they were at the beginning of the vote électronique trials.

4.3.1 Arguments in favour

Arguments in favour revolve around improving the voting experience so more voters can and want to participate. However, it is still unclear whether vote électronique will indeed increase turnout [131]. To save cost seems only be a minor motivation and indeed hard to tell, as long as economies of scale do not set in [38].

Besides the main arguments in favour presented afterwards, there are more which are not mentioned as prominently. Internet voting has to satisfy much stronger security properties than this is the case for postal voting (for example the verifiability properties). The voter can be prevented to submit invalid votes. Convenience increases, as the voter no longer needs to bring the voting letter to a post box or visit the ballot office. Using the interactivity possible through electronic means may enable a more satisfying and well-informed vote submission.

[25] https://e-voting-moratorium.ch

[26] Compare for example with the initiative text of the public initiative against vote électronique https://e-voting-moratorium.ch.

Expats In Switzerland, expats can vote on federal referendums and elections, can get elected and sign petitions, federal initiatives and referendums. Some of these rights can also be executed on cantonal level, depending on the canton. The number of expats is not negligible: In 2012, around 700k lived abroad, of which 125k were registered to vote [101].[27]

However, some are de-facto denied their voting rights as they are only able to use postal channel, and late deliveries are a reoccurring problem [211, 101]. The Organisation for Swiss Abroad (OSA) demands the introduction of an internet voting channel since more than a decade [101]. Indeed, when an internet voting channel was provided, the voter turnout of expats increased [130].

Inclusiveness Some voters are not able to fill out paper ballots independently. Computer-assisted voting might improve their voting experience and enable participation in elections without assistance endangering privacy or integrity of their vote. Indeed, voters with a disability are the prioritised target group, besides expats [36].

Societies representing blind people support vote électronique. Specifically, electronic voting allows the blind to fill out and send the ballot independently [51], hence guarding secrecy and integrity of the vote without trusting an assisting party.[28]

Long-term Viability In the future, voters may simply expect to be able to vote online, as all other contact with the administration [56]. Further, the postal channel will likely lose relevance over time and therefore also trust. This social change especially applies to young voters, but affects the whole population [191].

Working now on the next generation system is sensible as the roll-out will take many years (as it is indeed currently observed). As a comparison, the postal channel needed around 30 years to fully roll out, and is now used by around 90% of voters [191]. Electronic political participation is part of the digital strategy of Switzerland [125].

Save cost Current proposals still require to send secret material to the voter over post, but the vote itself can be returned over internet, saving significant cost and time. When a reliable secure channel is established to the voter (for example using an E-ID infrastructure), possibly even the remaining postal channel can be replaced. Further, the counting procedures scale better, which is especially useful in elections with many candidates and voting patterns.

[27] Note that these numbers are high enough to overturn national elections.

[28] Note that using a pencil or going to the postal office might not be possible without help.

In 2018, yearly cost of organizing elections is estimated at around CHF 13 for each eligible voter. With an electronic voting channel, the cost is expected to decrease around 30%. While this scenario assumes the voter still receives a letter, it does not include the fix cost for introduction and maintenance of the systems [38].

4.3.2 Arguments against

Arguments against vote électronique revolve around a perceived lack of security and the high complexity and cost of the systems.

Prominently missing from the arguments against vote électronique is fear of coercion enabled by voting at home, one of the main objections raised in international literature. This is not discussed in Switzerland as the issue would be similar with the already established postal voting, which is perceived as little impacted by issues like family voting [101].

Security The central argument against vote électronique is usually security. In Switzerland, notably the pirate party and the digital society are fundamentally opposed to vote électronique due to security concerns [51].

Legislation adapted often in the past decade, each time introducing much stronger security guarantees. Opponents of vote électronique doubt legislation and research is at a point where internet voting can be provided under acceptable risk. An additional danger, compared to the postal channel, is the perceived scalability of attacks: If an exploitable attack is found, it might scale much better than, for example, replacing letters.

The question of security is hard to answer, due to the complexity of the systems involved. Besides the valid arguments, many invalid ones or wrongly understood concepts spice up the discussions, sometimes leading to surprising change of minds.[29] Often the (invalid) comparison to e-banking is drawn, by both proponents and opponents of vote électronique.[30] The generally drawn comparison to postal voting might also not be helpful; both the accepted trust assumptions as well as the provided properties by the two approaches are very different, and therefore hard to compare.[31]

More nuanced, the arguments lamenting a lack of security can be categorized into conceptual and implementation issues. On the conceptual level, the current systems require full trust in the setup and printer component involved in preparing the election material, and trust in order to achieve

[29] Compare for example the Rat of Basel surprisingly abolishing vote électronique in February 2019 [261], and then taking back the decision already in June 2019 [260].

[30] Compare for example the discussions in Basel [261] and Zürich [241].

[31] Postal voting relies on trusted personnel, but has very weak formal guarantees. Internet voting relies on trusted authorities, but then reaches much stronger formal guarantees.

privacy in the end user device. These are strong trust assumptions, presumably because it is believed otherwise vote électronique could not be provided at all. On the implementation level, reaching the required properties under the trust assumptions is difficult and leads to high complexity. This impacts the quality of specification, proof, code and review of the systems.

Complexity The complexity of the current systems is huge. For a single vote of moderate size (for example, 4 referendums), the current systems execute a factor 10 more cryptographic operations as a normal TLS-secured web request would.[32] Further, some of these operations are very specialized constructions to resolve the verifiability-privacy conflict, sometimes used for the first time in industry, like zero-knowledge proofs, verifiable shuffles or oblivious transfer.

The size of the protocol, and the complexity of its operations, leads to complicated models and proofs. Finding issues is already difficult in small protocols, but even more so with multiple exchanges between different roles. An indication that this complexity is indeed an issue are the critical issues found when reviewing electronic voting systems (see section 3.2.3).

The specialized constructions for internet voting are not readily available but have to be implemented by the system provider from scratch. Implementation is not feasible without deep knowledge in cryptography, and experts in both cryptography and implementation are hard to come by. Further, the novelty and sparse usage of the constructions makes it relatively likely that new fundamental issues are still being uncovered and have to be patched. Both of these fundamental difficulties lead to critical vulnerabilities observed in the Swiss Post PIT 2019 [285].[33]

On top of the cryptographic primitives, the vote électronique system itself has to be added. Some non-trivial usability and technical challenges have to be solved that the system is understood and runs on every voter's device. Specifically elections, which leave much freedom to the voter about how they want to vote, are difficult to understand.[34] Further, the system has to interface to other governmental services (like the voter register), especially a challenge if the system should support multiple cantons.

Only a handful of experts worldwide are then fit to review the system due to the many internet voting specifics. With most researchers of the small field

[32] With over 150 operations (encryptions, hashes, ...) in CHVote and over 350 operations in the Swiss Post system for a reasonably sized vote over four referendums. Meanwhile, a single TLS connection requires around 10 operations for setup and execution.

[33] The broken verifiable shuffles implementation showed a lack of cryptographic understanding of the authors of the system. The weak Fiat-Shamir was known to be a problem in the literature since 2012 [25], but remained unpatched in the Swiss Post system.

[34] In some regions, over 25% of submitted ballots are invalid [81].

likely knowing each other, and some openly opposed to internet voting in general, independent and fair reviews are difficult to come by. Further, the reviews are somewhat constrained in time and scope, and arguably can not reasonably review the current systems in depth.[35]

Compared to postal voting, or voting in person, the process is therefore much more complex for all involved. This also complicates this voting channel from a democracy standpoint: It can be argued the voting process should be understandable to a large part of the electorate, clearly not the case with internet voting as conceptualized today. Further, organization of vote électronique is often done by the canton (due to cost, but also due to expert knowledge required), while elections traditionally used to be organized by the municipalities, leading to an increase in power concentration at the cantonal administration.[36]

Cost The cost of the already developed systems has been significant. The three first generation systems (until 2005) cost the federal government CHF 7.5 million, with additional expenses covered by the executing cantons. Afterwards, the federal government was not directly invested into development any more, but still provided coordination services to the cantons, for an additional CHF 2.5 million by 2012 [56].

Given 5.5 million eligible voters[37], each costing CHF 4 less when using the electronic channel [38], the yearly system cost must stay under CHF 22 million to not surpass the break-even. The Consortium system developed by Zürich cost CHF 11.1 million 2004-2011 [274].[38] The universal verifiable system CHVote cost CHF 4.5 million to specify and develop [265], with overall cost of 6.9 million for Genève [129]. Swiss Post does not publish official numbers [76], but according to internal documents, it spent around CHF 20 million so far (2021) [5].

Main cost driver is the complexity of the systems. The development of the central component is estimated at around CHF 2 million by Swiss Post, and at around CHF 1 million for the verifier component. It is argued that these components should be build multiple times by different vendors using different technologies to increase reliability [47]. Another cost driver is the often changing legal basis, which required continuous investment of the involved cantons. Swiss Post estimates the next foreseeable change in the protocol (relaxing assumptions on the printer and setup component) to cost

[35] For example in the Swiss Post system, the specification of the core protocol is 26 pages [235]. The implementation requires around 160k lines of Java plus other technologies [233].

[36] This change arguably contradicts the principle of subsidiarity - part of the constitution [105] - which states actions should be performed at the lowest governmental level sensible.

[37] 2020 average, https://www.bfs.admin.ch/bfs/de/home/statistiken/politik.html.

[38] CHF 2.3 million was covered by the federal government [274, 56].

CHF 1 million. The propagation of the change into the software is estimated at another CHF 1 million [47].

For an individual canton, these kind of sums are far from negligible. Since the initial three systems (for which the federal government supported the development to up to 80% [56]), no other canton has started another initiative on its own. While the three initial systems each survived around a decade, afterwards they were abandoned (Consortium, Genève) or handed over (Neuchâtel to Swiss Post). Cantons and civil societies repeatedly asked the federal government to either develop its own system, or again support cantons financially [130].

Given the large cost (and binding of other resources) relative to the number of primary benefactors of the system, some argue the effort is better spent on other projects of digitalization altogether. Alternative solutions to help the primary target group are proposed instead, like sending postal letters to expats earlier, or improving the assistance of people with limited mobility [130].

System-specific arguments The two remaining systems in Switzerland are very similar concerning trust assumptions, mechanisms and provided properties. However, their creation has been quite different. CHVote is based on a specification worked out in 2017 by the E-Voting group of the Bern University of Applied Sciences [139], and was then developed by Genève in-house [113]. The Swiss Post system was initially developed by Scytl, which operated similar systems also in Norway and Australia, repeatedly facing harsh critique for a perceived lack of quality of their solutions.[39] Swiss Post bought the system in 2020 to continue development in-house [232].

While Swiss Post is owned by the federal government, it is a private firm which primarily pursues economic goals. It is argued that this prevents full transparency [129], and that the people do not want central services of the state provided by private firms, as established in the 2021 E-ID referendum [51]. Further, economic incentives may prevent investment in quality efforts which cannot be observed or contracted by the customer. Indeed, observing quality of specification, proofs or code is arguably hard.

The past transparency efforts of Swiss Post have indeed sometimes been unfortunate. The source code was only made accessible as late as legally possible during the public review 2019, while the system was under control of Swiss Post already since 2015 [225]. Access to the source code had been hardened with restrictive usage conditions [45]. Then, the communication around the critical vulnerabilities found arguably downplayed their severity

[39] Compare with section 3.2.3.

or avoided acknowledgement.[40] Further, the code was perceived as low quality [45].

The new system which went into review in 2021 has more permissible usage conditions [233], and some components are provided under an open source license [237]. However, the main component is still proprietary [236], likely due to economic concerns [5]. Since 2019, the quality of the code has been invested in [233].

4.4 Administration

In Switzerland, elections are organized by the cantons, including for federal issues. While the cantons are mostly free how they organize their elections, they do have to follow some basic legal requirements. For vote électronique, this same principle holds: The systems are operated by the cantons, but the federal government defines some minimal legal requirements.

Developing such a system is costly, but once established, it could be reused by other cantons. The federal government decided to finance a large portion of the first three systems, developed until 2005, under the condition that the resulting systems could be reused by other cantons for free [55].

Since then, the federal government has restricted itself to facilitate between the cantons, but does not longer directly support development [56]. In 2020, the federal government still rejected developing a system by themselves, rather voicing support that different systems establish [129].

Zürich / Unisys (until 2015) Zürich had no centralized voting registers when the project started, hence harmonising these registers was a central part of the effort [32]. The resulting system allowed the municipalities to administer vote électronique independently (the so-called *Mandatenfähigkeit*). It was developed by Unisys [55].

The system then quickly rose to be the most adopted solution in Switzerland, thanks to the Mandatenfähigkeit. St. Gallen, Aargau and Thurgau used it to let expats vote, for which they form a logical municipality. Fribourg, Solothurn, Schaffhausen and Graubünden used the system as their municipalities administered their own voting registers [56]. Glarus joined some time later [58].

Besides adding support for elections and dropping the initial support for voting over SMS [56], the system remained nearly the same until it found

[40] For example, they proclaimed only low severity issues were found by the public introduction testing (PIT). This is technically true as the PIT did not include the review of the source code (where the severe issues were found), but in the public this was often perceived as the same thing, which they even acknowledged in their own report [228].

its end in 2015, when the federal government denied continued usage: The system had an unresolved issue jeopardizing vote secrecy, and did not comply with the new regulatory framework established two years before [58]. The involved cantons then gave up the system [289].

Genève Genève already had its voting registers centralized when it started development in 2001 [32]. The security was provided by blue-infinity, a firm situated in Genève, which was also asked for reviews by the other pilot systems [55].

Starting 2005, the system was fully developed in-house. Genève continued development, and continuously improved usability and security [56]. It was adopted by Basel-Stadt, Luzern and Bern [113, 58].

By 2015, the system supported individual verifiability [58]. In 2016, it was decided to make the systems open source and develop a new version which also provides universal verifiability [113]. In 2018, the development of the new version was halted due to its high cost [114], and the system providing individual verifiability was shut down in 2019 [115].

Neuchâtel Neuchâtel added the internet voting capability into Guichet unique, an online portal which also implemented other electronic administrative concerns [32]. Similar to Genève, they additionally took an effort to centralize voting registers and election processes [55].

The cryptography was provided by Scytl. A form of end-to-end encryption was already implemented, as well as the confirmation/finalization code mechanism still in use today. Two different audit rounds discovered security flaws, which were fixed before the first official election [55].

The system was not used by other cantons, presumably because the integration into the Guichet unique was very tight and could not easily be adopted to other cantons. Neuchâtel however still invested into further development, adding elections by 2011 [56], and providing individual verifiability by 2015 [58].

Starting 2016, the system was operated by Swiss Post [225, 226], which then also provided it to other cantons. By 2019, Freiburg, Neuchâtel, Basel-Stadt and Thurgau used the Swiss Post system [144].

However, the Swiss Post system failed the public review in 2019, and had to be taken offline [41]. Swiss Post bought the rights of the system from Scytl in 2020 [173], and from then on continued development in-house [232]. The updated system went in review mid-2021 [48], and is the only remaining system in Switzerland.

4.5 Summary

The history of vote électronique is first a slow, but a successful one: Three competing systems, with slightly different approaches, enable half of cantons to provide vote électronique within a decade [56]. However, then the legal requirements become stronger, leading to the Consortium system to give up [58, 289]. And shortly before introducing vote électronique as a regular voting channel, the other two remaining systems are shut down; the Genève system due to cost [114], the Swiss Post system due to critical vulnerabilities [41]. By 2022, Swiss Post is the only provider which aims to bring back vote électronique in the foreseeable future.

The legal requirements have continuously been hardened throughout the project, and have undergone three consultation procedures within the last decade [33, 46, 52]. Besides enforcing overall high quality and transparency of the systems, specifically required are individual verifiability, universal verifiability and vote secrecy, albeit under quite strong trust assumptions.[41] The future development of the law has already been hinted at in a report of 2020: The security assumptions in the setup and the printer authorities should be lowered, towards verifiable parameter generation and distributed printing [47]. In general, the legal provisions are very detailed with the central requirements comparable to the recommendations of the Council of Europe.

Politically, not much has changed since the start of the project, although much more interest arose when Swiss Post suddenly was the only provider left, and their system featured critical vulnerabilities. The pro arguments revolve around the inclusion of more voters - specifically expats and voters with disabilities - and porting voting to the internet as done with other governmental services. The contra arguments argue with too high cost and complexity of the systems, while still being insecure. At least the contra arguments have shifted somewhat, as with each revision of the law towards stronger security guarantees, cost and complexity have risen significantly.

Three cantons, with initial support from the federal government, started with their own voting system, with diverse approaches for architecture, transparency and security. Over time, all canton-lead initiates came to an end, as either they could not keep up with recent developments (Consortium) or the systems became too expensive (Genève, Neuchâtel). The remaining system was initially developed for Neuchâtel, but is now marketed towards all cantons by Swiss Post.

[41] Full trust in the setup and printer authorities and trust for secrecy in the voter's device.

Part II

Proposal

Chapter 5

Setting of vote électronique

We explore in this chapter the setting vote électronique is in, hence which requirements it has to implement under which guarantees.

The requirements are fortunately straight-forward. An electronic voting system for Switzerland should enable eligible voters to participate in referendums and elections. Out of scope for this work is anything which does not impact the cryptographic protocol, like user interfaces or import and export functionality. We however of course keep in mind what has to be built on top of the cryptographic protocol, and might choose an appropriate trade-off.

Concerning the guarantees we have to deliver, these are described by the law. This work focusses on the cryptographic protocol of a vote électronique system, hence we are most interested in the technical details described in VEleS and its appendix [35], but not so much in the organisational constraints regulated in VPR [53, Art. 27] or even the more abstract BRP [98, Art 8a]. We use the newest version of VEleS currently available which was published as part of the consultation procedure in summer 2021 [50, 52]. It will be set in force sometimes in mid 2022 [60], possibly featuring some small changes.

Given the requirements and the legal framework, we then discuss how we might change the setting. We might be able to further minimize trust assumptions or even to strengthen requirements. Further, we explore ways how to reduce the complexity of a possible vote électronique system, as we note that the existing systems are very complex, and this complexity impedes the operation of a cost-effective and high-quality system.

Referendums, elections and eligibility Referendums are around four times each year, and are primarily a yes/no choice. Abstention is also possible. Further, some motions have a direct counter-proposal worked out by the parliament; in which case both proposals are individually voted for or against, and additionally a preference between the two is specified.

Elections are every four years, leaving much more freedom to the voter with modifiable lists of candidates. We omit describing here in more detail the different ways these lists can be modified as it will not be relevant for the protocol proposed as part of this work.

Eligible voters are organised in voting circles. Smaller municipalities may consist out of a single voting circle, while bigger municipalities may consist out of many.[1] The results are announced at the granularity of voting circles. Within a voting circle, there may be voters of different eligibility.[2]

5.1 Legal Framework

For now, we keep the roles and properties somewhat imprecise, and align the terminology and formulations with law [52] rather than literature.

Besides the model and the properties we describe here, the law sometimes imposes artificial constraints on the protocol. It is our understanding that these constraints are accidental (for example, when mentioning public and secret keys, the law arguably implicitly requires the protocol to indeed use public and secret keys), hence we will not elaborate on these further.

Constraints The legal framework for vote électronique is developed within some legal, political and administrative constraints.

The cantons are in charge of organizing the elections, and - by the Swiss federal structure - in principle free to do this as they please. The federal government is however able to constrain their options (for example, by passing a law which defines minimal security requirements), but will be careful to only regulate what is strictly necessary.

The postal channel is currently the only feasible way to send secret material to voters. There is no public-key infrastructure with citizens, and an E-ID which aimed to bring something similar, was rejected in March 2021 [278].[3]

The postal voting channel, and the in-person voting channel, are already established and it is currently not the aim to replace these channels [37].

Cantons are organized differently, some tend to place responsibilities at the cantonal administration and some distribute more to the municipalities. Whether it makes sense to include the municipalities in vote électronique

[1] The city of Zürich consists of 12 voting circles.

[2] For example, foreigners may be able to vote on cantonal issues but not federal.

[3] The rejection was however arguably not because citizens do not want a state-recognised electronic ID, but rather because the implementation made it possible that private firms could have been able to provide it (see https://www.e-id-referendum.ch/).

depends on the canton.[4] For centrally organized cantons such as Basel-Stadt and Genève, appropriate structures on municipal level may not exist (or be unrealistic to use for this purpose). In other cantons, for example St. Gallen and Thurgau, an inclusion of the municipalities would follow the established strategy (as supported by the Consortium system used until 2015), and may even be politically required.

Voters do not have access to a trusted device, and are intellectually limited. As the voting channel should be fit for use for a large portion of the electorate, we cannot ask the voter to do complex tasks like keeping secret key material safe over a longer period of time, or executing more than very basic[5] mathematical expressions.

5.1.1 Model

The legislation assumes a certain model when describing the security properties.

Roles The protocol is initialized by the *Setup Authority* which receives all data related to the election (eligible voters, voting options, deadlines, ...) and uses it to prepare the election. The *Printer Authority* is then able to transmit data securely to the voter (over the assumed secure postal channel) following the instructions provided by the setup authority.[6]

The *Voter* casts and confirms their vote using the *Voter Device*. Voters are additionally allowed to invalidate their vote by claiming it has been received unsuccessfully by the servers [52, appendix 4.9].[7]

The *Control Components* may participate in all phases. Multiple of these components are run in parallel, and together they ensure the voting system behaves as expected.

The *Auditors* observe the voting system for correct behaviour (but do no actively participate). They are able to impersonate fictional voters to test the system [52, appendix 14.7].

Covert Adversary The law assumes covert adversaries.[8] Covert adversaries, also called honest-but-curious adversaries, may deviate arbitrarily from the protocol, but do not wish to be "caught" doing so [12]. This attacker model notably excludes many disruption of service attacks (like a

[4] Note that the federal government will likely not regulate this, as arguably this would restrict the cantonal autonomy too much.

[5] Like comparing values, or addition of small numbers.

[6] The split of these authorities likely has practical reasons: A facility optimized for printing might not be suitable to prepare the election material.

[7] How the claim is authenticated and implemented is outside the scope of this work.

[8] Notably, CHVote [139] and the Swiss Post protocol [235] come to the same conclusion.

component refusing to process input, or an accountable component flooding others with requests).

Trust assumptions Full trust is placed into the setup and printing authority, hence both are assumed to operate correctly and without leaking secrets. These authorities are only active in the setup phase of the protocol, before the voting phase starts.

Some voters might be dishonest, but for honest voters, the to-be-achieved properties must hold. Voters are only active during the voting phase.

There are multiple control components and auditors (the law suggests at least four of each), but only one each is fully trusted, while it remains unspecified which one. Both control components as well as auditors are potentially active during all phases of the protocol, including the tally in the end.

The roles are connected to each other over bidirectional untrusted channels. The auditors may not send messages.[9] The printer authority may also not send messages, except a postal letter to the voter, which is assumed a secure channel. The second assumed secure channel is from the setup authority to the auditors.

5.1.2 Properties

Having set the stage with the involved roles, we now define list the properties as defined by the law. As in the literature, there is so far has no consensus over definitions of the security properties, hence it increases transparency if the law explicitly defines the intended meaning. We formalize the properties in section 8.2.

Following legislation, we refer to *Complete Verifiability* if both individual verifiability as well as universal verifiability are guaranteed. A vote is *registered successfully* if the server acknowledges that it is the first vote cast and confirmed by the corresponding voter, and that the selection of voting option(s) it represents is valid.

Individual verifiability Individual verifiability holds when voters are given exactly one of two proofs: Voters who participate electronically are given a proof that the vote has been registered successfully by the server, exactly as cast. Voters who did not participate electronically can request a proof that their vote has not been registered by the server [52, article 5.2, appendix 2.5].

[9] This essentially means that the auditors cannot be actively involved in the protocol, hence we never include them in the protocol description. Auditors may nonetheless observe and validate progress from the outside.

Voters check their proofs independently. Security analysis may assume the printer and setup authority as well as one control component are honest [52, appendix 2.9.1].

Universal verifiability Universal verifiability holds when the auditors are given a proof that the result is composed out of all, and only of, successfully registered votes [52, article 5.3, appendix 2.6].

Auditors check the proofs using technical aids. Security analysis may assume at least one auditor (with their technical aid), as well as at least one control component are honest [52, appendix 2.9.2].

Vote secrecy and Fairness Vote secrecy holds if the plain vote cannot be attributed to the voter. Fairness ensures the attacker does not learn partial election results before the official tally [52, article 7, appendix 2.7].

Auditors check the proofs using technical aids. Security analysis may assume at least one auditor (with their technical aid), the printer and setup authority, as well as one control component are honest [52, appendix 2.9.3]. The law further allows the voter device to be honest for this purpose, hence not leak any secrets. It is implied by the law, although not clearly specified, that the short ballot assumption holds.[10]

Authentication Authentication holds when the attacker cannot insert votes without having control of the voter [52, appendix 2.8].

Security analysis may assume the printer and setup authority as well as one control component are honest [52, appendix 2.9.4].

5.2 Discussion

We explore how we may optimize the setting, under the given constraints. Our observations are independent of any protocol, and we argue for universally true baselines. The insights are not formally established, but nonetheless will help us design our protocol: They prevent us thinking about mechanisms which are fundamentally impossible.

There will likely still be a gap between our concrete proposal and the minimal trust assumptions or the strongest possible properties established here. Such a gap may reflect some unidentified fundamental restriction we have overlooked, and might not be resolvable. Or the gap is chosen deliberately, as we resolve tradeoffs between higher complexity or higher security towards simplicity.

[10] Hence just seeing a list of plain votes does not allow to attribute the respective voters.

5.2.1 Minimize security assumptions

We first look at the minimal trust assumptions towards the authorities. We establish what their fundamental non-transferable tasks are, and then conclude how this impacts the trust assumptions.

We find that the setup and the printer authority require very strong assumptions in order to achieve our security properties. Introducing a second printer authority to relax the trust assumptions would unfortunately not bring much benefit, but increase the effort to operate the protocol substantially. We also argue that having at least one trusted control component and auditor each is a trust assumption which cannot be reduced any further.

Where we however see potential to reduce the trust assumptions as assumed by the law is with the voter device: By placing the (arguably realistic) burden of encrypting the vote on the voter, it is possible to place no trust in the device at all.

Relieve the setup authority The setup component is where the election authorities enter all data related to the election (eligible voters, voting options, deadlines, ...).

We do not need trust in the setup authority for any data entered which is public (like deadlines, length of cryptographic parameters, ...). This data can simply be published, and for example auditors can verify its correctness. Further, we do not require trust in the setup authority to generate the cryptographic key material. This can instead be done with the help of the control components, using some sort of multi-party computation scheme.

However, there are two tasks we do need trust into the setup authority for. This includes data that needs to stay secret and/or cannot be publicly verified (like personal data of eligible voters), and data required to prepare the data for the printing authority. The former implies trust at least for eligibility verification[11], the latter implies that the setup authority inherits the trust assumptions of the printer authority.

Harden the printing authority The printer authority prints out data received by the setup authority, and sends it over the postal channel to the voter.

In our model, this printed ballot sheet is the only form of secure communication with the voter before the election. This implies that the printer authority learns all secrets the voter receives, and therefore can break authentication (both for the server authenticating the voter as well as vice versa). As we

[11] This is a part of authentication, see section 8.2.

need to trust the printer authority to not break authentication, we also need to trust it for individual verification, which depends on authentication.

When the encryption scheme used to encrypt the vote is deterministic, we additionally require trust in the printing authority in order to achieve privacy. Instead, a randomized encryption scheme could be used, which however would imply trust in some technical aid to pick good randomness.

Introduce secondary printing authority To circumvent a single authority knowing all secrets a voter receives, we can introduce a second printer authority with another secure channel to the voter.

This may enable us to avoid trust into the printing authority in order to achieve authentication, rather transforming it into a non-collusion assumption between the printers. The voter and the server would essentially be required to verify two sets of authentication secrets, with each set originating from a different printer authority.

For privacy, when some form of deterministic scheme is used, a similar insight applies. The voter would again need to somehow combine information received from both printers into a single vote.

This second printer authority would need to have different organisational structures as the primary printing authority to achieve an actual increase in security. For example, one of the printers could be operated by the canton, and the second by the municipality.[12] However, this will increase the effort substantially (at least doubling it), which is even more relevant as the printers still have to be operated under strong procedural safeguards.

To reduce the effort somewhat, one could design this secondary printing authority such that it delivers secrets valid for multiple elections. For example, when registering for vote électronique, the voter receives a code book valid for the next two years. However, this now requires the voter to safeguard these secrets. Voters may lose the key material, or choose unsafe ways to store it.[13] Further, some printing authority has to hold on to these codes over an extended period in time, which may require additional trust assumptions or procedural safeguards.

At least one trusted control component The control components are able to contribute throughout all phases of the protocol.

As noted when discussing the setup authority, the control components may generate the cryptographic key material used for the election. This can be

[12] However, depending on the canton, appropriate municipality structures may not exist. Further, some cantons form a logical municipality for expats (with no real-live equivalent).

[13] Note that the key material is used only every three months.

designed in such a way that only a single honest control component ensures the material has the required properties. For example, when combining multiple values using XOR to get a uniform at random key, we only need a single of the original values to be uniform at random for the result to be. Similar results are possible with more complex schemes such as public key aggregation or Shamir secret sharing.

During the voting phase, the control components are the only authorities the voter communicates with, hence also the only components that store the votes. It requires at least one honest server to ensure received votes are not simply dropped. Note that this also requires each individual server to authenticate themselves to the voter, so the voter can be sure the honest server has seen their vote.

In the tally phase, the control components may again be involved. The tally could be constructed in such a way that its operations are provable. A single honest control component would again be enough to uncover missing or wrong proofs.

At least one trusted auditor The auditors are active during all phases of the protocol, but do not actively participate.

At least one auditor has to be trusted, else the audits would serve no purpose (the attacker could simply claim the success of the audit independently of the input).

Interestingly, there is no inherent need to trust any auditor to achieve privacy. Whether the encrypted vote conforms to the intent of the voter has already been checked by the voter (due to individual verifiability), therefore this does not need to be checked again by the auditors.

No trust in voting device The voter device processes the input from the voter and communicates with the control components.

The voter clearly requires some technical aid to send a request over the internet. However, the voting device's function can be restricted to that of a gateway between the voter and the servers, simply forwarding data. It only sees obfuscated data, and the forwarding is strictly checked. Then there is no need to trust the voting device in order to achieve any of the security properties.

For sending data from the voter to the servers, the voter can use voting codes instead of plain values. For this to work, the voter and the tallying authority need to agree on the same *plain value* $\leftrightarrow$ *voting code* lookup. When the device does not know this association, it learns nothing when snooping the traffic.

To ensure the data was indeed forwarded to the server as intended, the user needs some verification. This can be solved by placing another lookup both

at the voter and the server, concretely *voting code* $\leftrightarrow$ *verification code*. If the voter receives the correct verification code back from the device, it knows the forwarding was successful. This of course under the condition that replays are impossible.

Compared to the current systems, voting codes are a new proposal. Likely, it was not considered so far as the usability when entering the vote decreases.[14] However, understanding of the system as a whole might in turn increase: What the voter enters is exactly what the server receives (rather than being obfuscated by the voter device). Further, the task is in is essence a simple one: The voter needs to perform a lookup, something they already do in reverse for the established verification codes. It should be feasible to design appropriate usability measures to make this work.

5.2.2 Strengthen properties

We now look at the properties required by Swiss law, and argue whether and how they could be strengthened.

For privacy, we conclude that the required properties are as strong as they can be under the given setting. Notably the stronger notion of ballot privacy called receipt freeness contradicts individual verifiability directly. Everlasting privacy, while not strictly impossible, seems unreachable from a practical point of view.

For verifiability, we argue that the most important properties from literature are indeed already required. Additionally, accountability or the even stronger dispute resolution properties could be required. These properties would improve the availability guarantees, which could then help to assume a stronger attacker (rather than the covert adversary model, which essentially is assumed to not break availability). However, these properties are rather new in the literature, and their implementation would likely entail a lot of additional complexity.

Privacy Swiss law requires vote secrecy (no one learns the vote of some specific voter) and fairness (no one learns votes until tally). These are the oldest recognised properties in the literature, and virtually any voting system fulfils them (including non-electronic voting).

Stronger than vote secrecy, but usually seen as a baseline in the literature, is receipt freeness: The voter should not be able to produce a proof of how the vote was submitted, even when collaborating with the attacker, to prevent

[14] Indeed, in a meeting with the Federal Chancellery on 24.11.21, this was the raised doubt, while the security benefits are recognised and seen as important. In a meeting with the cantonal administration of Basel on 21.10.21, the feedback was carefully positive.

vote selling. For in-person voting, this is usually guaranteed by the voting booth, for the Swiss postal voting however there is no such guarantee.

In the vote électronique setting, any secret information the voter has is contained on the ballot sheet sent by postal mail. As we must assume the voter collaborates with the attacker, and forwarding the ballot sheet to the attacker is clearly practical, we must assume the attacker has the same knowledge as the coerced voter.

If the attacker coerces the voter before they hand in the vote, we cannot have individual verifiability at the same time as receipt freeness. The attacker will be able to observe the individual verifiability mechanism (for example by shoulder-surfing), and as the attacker has the same knowledge than the voter, the mechanism must also convince the attacker of the cast vote, which breaks receipt-freeness. The only way to recover would therefore be mechanisms that break individual verifiability, like declaring a confirmed vote as invalid by calling an election hotline.[15]

But what happens when the attacker coerces a voter only after the vote has been cast?[16] When the individual verifiability check can be repeated, or when by some other way the actions of the voting device can be retraced[17], receipt freeness still breaks. It is unclear whether a protocol exists under minimal trust assumptions which defends against both these threats.[18]

We omit a separate discussion for coercion resistance, as it is a strictly stronger property than receipt freeness.

Everlasting privacy, hence not relying on cryptographic assumptions for the privacy of the vote, is arguably possible. Corresponding cryptographic primitives are well known, like perfect secret encryption or perfectly hiding commitments. Other cryptographic operations could be shifted into the real world, like performing a verifiable shuffle using paper. Unfortunately, the practicality of especially the latter example is doubtful, while the verifiable shuffle is a seemingly necessary building block of an internet voting scheme.[19]

Verifiability Swiss law requires individual verifiability (the voter can check the vote was received by the server) and universal verifiability (it can be audited that any and only received votes are included on the tally), and

[15] Or revoting, but this is not possible in Switzerland.

[16] Note that once the postal vote envelope is sent, the voter can no longer proof their vote.

[17] Note that universal verifiability guarantees cast votes can be accessed.

[18] In the Swiss Post protocol, the encryptions are deterministic, hence given the submitted ciphertexts and the secret key (from the ballot sheet) the original voting choice can be recovered [235]. CHVote uses randomized encryption, but consequentially requires trust in the voting client for the randomization [139].

[19] See the discussion in section 6.2.2.

refers to complete verifiability when both are fulfilled. Swiss law further requires some notion of an authentication property, which is best described as a precondition for individual verifiability, and the notion known as eligibility verification in the literature.[20] All these properties are widely regarded as necessary in the literature.

End-to-end verifiability is not required by the law, although argued by most researches as another baseline. However, individual and universal verifiability together (which are indeed required) almost always[21] imply end-to-end verifiability [82].

One could think about strengthening eligibility verification, explicitly allocating each vote to a voter publicly (e.g. how [167, 252] implement it). It is clear from a data protection perspective that in a real system the clear name cannot be used, but if only an alias is chosen instead, the benefit from a practical point of view is arguably low. But then who creates this alias list, and how does this actually increase the trustworthiness of the system?

Accountability ensures that actions of authorities are attributable, so dishonest behaviour is not only detected, but also attributed to a specific offender. It seems realistic that this property can be reached.[22] The stronger dispute resolution property would additionally required complex secret distribution mechanisms to then additionally recover from a misperforming party. Both properties have not received much attention in the literature yet.

In the context of Switzerland, both accountability as well as dispute resolution could be seen as an extension of availability. As a any more adversary is assumed, breaking availability is consequentially out of scope. However, if increasing accountability is possible, while not jeopardizing more important properties or increasing the complexity too much, it would clearly strengthen the system.

5.2.3 Reduce complexity

Given the security properties, and the trust assumptions, we explore options how complexity can be taken out of the system.

Complexity indeed seems to be a big issue, raising to a point where acceptable cost and quality is difficult to obtain. Swiss Post had quality problems [45], although it spends a small fortune on its system [5].[23] CHVote

[20] See section 8.2 for a formal discussion.

[21] With the exceptions arguably only applicable to exotic voting schemes.

[22] For example, all messages from authorities are signed, and when an invalid message is singled out, the offender could be clearly blamed.

[23] The timeline of the sources are skewed: The quality problems were discovered in 2019, but the report that Swiss Post spent a lot is from 2021. The argument still stands, as it is unlikely Swiss Post spent most of the money 2019-2021.

was specified only in 2017 [139] by experienced researchers but used similar mechanisms and ended up being similarly difficult to understand as the older Swiss Post system.[24] While cost was around a third of what the Swiss Post system presumably cost[25], it was still a significant amount and too much for a single canton to bear [114].

We can reduce the complexity by choosing simpler and industry-standard cryptographic primitives to avoid difficult implementations on our own. We can focus to implement first and foremost referendums, to avoid the additional complexity introduced by the complex voting procedures of elections. Further, we can externalize functionality by letting the voter or some other system execute it, like encryption or shuffling.

Use simpler and industry-standard cryptographic primitives Some of the used constructions are very complex, and implemented for the first time by industry. These complicate the implementation (as it has to be done from scratch) but also the review (as only domain-experts understand the constructions in depth). Using simpler cryptography is clearly beneficial.

However, these constructions are complex because the system has be verifiable, hence a third party has to be convinced that the system executed correctly. Sometimes this verifiability has to be proven even while preserving privacy, hence a third party has to be convinced that the system executed correctly *while not learning any of the secret material*. This complexity is somewhat inherent to electronic voting (like the tally phase, which has to count correctly while not leaking the plain vote of individual voters), but there are many different ways how to achieve it.

For example, zero-knowledge proofs of knowledge can be used to assert some value has been correctly re-encrypted, without revealing the plaintext or the secret key. Depending on how the protocol is setup, this same assertion can be reached by taking advantage of time: The re-encryption is done, and the input and output of the operation (but not the secret key) is published. After some time, when the secrecy of the computation is no longer relevant, the secret key is also revealed. Then anyone can verify the computation was done honestly.

Similarly, commit schemes can be implemented using hash functions. Key material from different sources can be combined using XOR. Verifiable encryption can be done by the users themselves using a lookup of precomputed values. Public key cryptography can be replicated with a hash function and an additional integrity assumptions into a third party.

[24] Fewer, but more uncommon, cryptographic operations are used (like oblivious transfer).

[25] 6.9 million for Genève [129] vs 20 million for Swiss Post [5]. For the Swiss Post system, this number likely does not include much of the cost borne by Scytl.

Reduce functionality Besides restricting interoperability features (like import and export interfaces) and similar technical depth, we may also restrict functionality visible to the voter. Keeping the requirements down of a system arguably reduces development time super-linearly (meaning, half the features likely use less than half the development time).[26]

A likely candidate is to drop support for either referendums or elections. Focusing on referendums first is in line with the initial strategy already proposed 2002 [54] but a step back compared to the current systems which at least theoretically support elections. As the legal requirements and the technical implementations have changed heavily in the last decade, it can be argued that also a restart from a requirements point of view is sensible. However, voters might wonder why the tool is only sporadically available, and the uncertainty might lead to distrust in the overall system. If only partial support is possible, voters should be clearly informed about it (for example by calling the system explicitly "votation électronique").

Externalize tasks Executing tasks outside of the system, rather than doing it within, may also reduce complexity. A vote électronique system reduced to its essential features (allowing the voters to vote) has not many tasks left to take out. However, even cryptographic operations can be externalized.

The cryptographic shuffle has a physical counter-part, concretely printing sheets then physically permuting the order. This is already implicitly in use with ballot boxes, where filled out votes are put in by the voters. The order of insertion is lost, thereby losing the association between the voter and their vote. To ensure that this shuffling cannot be inverted, no attributable information must be contained on the permuted sheets.[27]

The cryptographic encryption can be delegated to the voter, too. The voter can be presented a voting code for each plain vote, and the voter uses said voting code to communicate their intent. This is already in use - the other way around - with the verification codes known in the current systems. Note that the voter essentially performs a lookup of a precomputed cryptographic operation, and whatever creates or sees said lookup has to be fully trusted.

More complex cryptographic operations may be executable on a trusted voter device. However, in Switzerland no smartcards or similar are in widespread use, hence an approach as for example used in Estonia (see [148]) is not feasible.

[26] Intuitively, one can argue as follows: Given three components, and a forth one is added, we need to add three interfaces to connect it to the other components. We hence double the amount of interfaces required to six, while only increasing functionality by a third to four.

[27] We omit here how ciphertexts might also enable re-attribution.

5.3 Summary

We have seen the model and security properties as given by the law. After establishing terminology, the law clearly states trust assumptions for the roles participating in the protocol. We identify the covert adversary model as the attacker the setting attempts to defend against. The law further defines the security properties a vote électronique system must fulfil.

We then discussed the current setting of vote électronique, and identified some small possible improvements. We further minimize trust assumptions by arguing that the voter device needs not to be trusted. Further, we have seen that accountability and dispute resolution are not reflected yet in the law, but also acknowledge these contribute primarily to the availability of the system.[28] Also, we have motivated that reducing complexity is a necessary improvement, and have discussed strategies to do so.

[28] Which is assumed not be of primary concern as the attacker model shows.

Chapter 6

Iterative Proposal

With the setting established and some general improvements suggested (see chapter 5), we may now start to propose our protocol.

Instead of starting with a full and formal description (which is in the next chapter 7), we want to introduce the protocol iteratively. This aims to ease understanding of why each specific messages is necessary, but also serves as an (informal) optimality argument: Given the same setting, each message of our protocol is required, and therefore the protocol is as simple as possible. Given the aim of this chapter, we focus on the mechanisms used but forego precise mathematical definitions.[1]

We foremost target simplicity in the voting phase (rather than in the setup or tally phase), as this is where the voter interacts with the protocol. We understand that a protocol appears simpler when there are few single-purpose values and few round-trips. Our strategy is as follows: We focus first on the voting phase, and establish two separate protocols, one fulfilling the privacy properties, and the other fulfilling the verifiability properties. Then we join the proposals together, and define suitable setup and tally procedures.

Simplifications We assume voters of different eligibility and voters of different voting cycles to participate in different elections. This simplifies our protocol description. In practice, the protocol can then be run multiple times in parallel to serve voters of all eligibilities and voting cycles.

Bulletin Board Besides the roles introduced by the setting in chapter 5, we also rely on a *Bulletin Board* (BB): A publicly accessible append-only storage primarily for the exchange between the voters and the control components. Each addition to the bulletin board is confirmed with a signature over it sent

[1] Readers which prefer precise mathematical treatments may prefer to skip ahead to chapter 7.

to the control components. The bulletin board is the only component publicly reachable, and therefore the only component which requires hardening like public servers do (for example, against DDoS attacks).

Given the covert adversary model of our setting, we do not require any trust assumptions in the bulletin board. It will only append valid messages, which for the adversary are either hard to guess (because they require authentication secrets) or attributable (because they originate from a control component or an authority). Further, the bulletin board cannot refuse to append messages, because again this would be easily detectable. As the bulletin board state is signed by the bulletin board itself, it can never remove messages (as a signature would proof misbehaviour). Providing different parties with different views is also easily detectable, and therefore not part of the adversary model, either.

Notation We use boldface for an unordered list of elements, for example $\mathbf{a} = [a_1, a_2, ...]$.

We denote the combination of permutations using $*$, like $[0 \to 1, 1 \to 0] * [0 \to 0, 1 \to 1] = [0 \to 1, 1 \to 0]$. We overload the $*$ operation to also query lookups[2], like $a * [a \to 2] = 2$. We may also revert lookups, like $2 * [a \to 2]^{-1} = a$.

We denote modular addition with $\oplus$. The modulo is defined over the group from which the individual values have been drawn from (for example if $x, y \xleftarrow{r} \mathbb{Z}_v$ and $\mathbf{L} = [x, y]$, then $\oplus \mathbf{L} = \sum \mathbf{L} \bmod v$).

For partial values of the control components we use lowercase with an index top right (for example, $vv^{(i)}$ for the partial vote verification of $CC^{(i)}$). When the partial values have been combined, we use the uppercase equivalent (for example, VV for the vote verification).

6.1 Establish verifiability

First, we establish verifiability. We aim to find a protocol which supports vote submission and keeps the vote safe and unmodified until tally.

As we target here only the voting phase, we assume for now *OptimalSetupTally*: The setup and tally procedures are defined in such a way that they do not jeopardize the properties achieved in the voting phase. This notably includes that the secrets are distributed appropriately.

[2] We refer to lookups for permutations where the source and target groups do not match.

6.1.1 Verifiable internet voting

We start by the voter simply sending the vote authentication VA together with the plain vote $\mathcal{P}$ to the bulletin board. The bulletin board accepts the vote if the VA is valid. To enable the bulletin board to check whether VA is valid, but without giving it the ability to impersonate a voter, we let it know hashes of all valid VA.[3]

For the bulletin board to correctly receive the vote and then keep it, we need some simplifying assumptions. We assume *SecureCommunication* hence messages sent cannot be read or changed by the attacker.[4] Further, we require both *DeviceIntegrity* and *BBIntegrity* such that the messages are forwarded and kept unmodified. We will replace these assumptions with weaker ones that conform to our target security model over the next iterations.

With these trust assumptions, we already reach all required integrity properties: Complete verifiability is reached as the voter can check both his own vote, as well as all other given votes on the bulletin board. Authentication works as the attacker only learns the one-time-use VA after it has been used.

The voter **V** knows the vote authentication VA and the plain vote $\mathcal{P}$. **V** uses their device **D** to send $(VA, \mathcal{P})$ to the bulletin board **BB**.

$$\begin{aligned} \mathbf{V} &\rightarrow \mathbf{D}\colon\ VA, \mathcal{P} \\ \mathbf{D} &\rightarrow \mathbf{BB}\colon\ VA, \mathcal{P} \end{aligned}$$

Assumptions: OptimalSetupTally, SecureCommunication, DeviceIntegrity, BBIntegrity
Properties: Complete Verifiability, Authentication

Protocol 1: Verifiable internet voting.

6.1.2 Fewer integrity assumptions

We now drop the unrealistic and unlawful trust assumptions by introducing the control components.

We assume that at least one control component is honest, calling this the *SingleCCHonest* assumption. This already avoids any trust assumption in the bulletin board, as the append-only guarantee is now ensured by the control components.[5] Further, we need an *ElectionHotline* assumption; an external

[3] By the OptimalSetupTally, we do not need to specify now how this exactly works.
[4] This also holds for the attacker controlling the bulletin board.
[5] See the discussion at the beginning of this chapter.

trusted authority which is able to mark votes as invalid if the respective voter complains.[6]

We further introduce a vote verification mechanism using the control components. In the setup phase, each control component i chooses for each $(VA, \mathcal{P})$-tuple a uniform at random partial vote verification $vv^{(i)}$. During the voting phase, they publish $vv^{(i)}$ when the corresponding $(VA, \mathcal{P})$-tuple appears on the bulletin board. Honest control components will never publish multiple $vv^{(i)}$ for the same VA, ensuring for each voter only a single vote is verified.

The voter then sums up[7] all the partial vote verifications $VV' = \oplus_{i=1}^{m} vv^{(i)}$, and checks whether $VV' = VV$ for VV the actual expected vote verification for the submitted $\mathcal{P}$. If the check succeeds, the voter can be reasonably sure the $\mathcal{P}$ was received by all control components. If the check fails, the voter calls the election hotline to complain.[8]

Even though we changed the trust assumptions, the resulting properties remain the same. We achieve individual verifiability by the voter checking the verification codes. Universal verifiability is reached as before; any submitted votes are visible on the BB. Authentication works as the attacker only learns VA when the voter submits its vote, and would then break the individual verifiability check if it modified the vote.

6.1.3 Automate voter feedback

The voter feedback whether the individual verifiability check succeeded currently requires a hotline to receive a call whenever the check failed. Depending on the number of disruptions, this may be a lot of effort to operate. We therefore introduce a second round-trip to automate the feedback of the individual verifiability check. This indeed brings an advantage on the assumption that an attacker is less likely to interfere when it cannot change the cast vote.[9]

If (and only if) the voter has received the correct vote verification, they enter a confirmation authentication CA. A vote is only included in the tally if the CA is published. Similar as with the vote verification, we establish a confir-

[6] Correct authentication and implementation of the claim is included in this assumption.

[7] Using modular addition, according to the group the vv were picked from. Note that XOR would have the same effect if the values are picked appropriately.

[8] Note that the expected VV' only results with high probability if all honest control components have seen the same $(VA, \mathcal{P})$ combination. If an honest control component i has seen a different $(VA, \mathcal{P})$ combination, the attacker has no advantage to guessing the $vv^{(i)}$.

[9] In the real world, an additional assumption can be taken: If the attacker could not interfere with the first round-trip, it will not interfere with the second (the attacker might be inactive at the moment, might have been unable to capture the connection, ...).

The voter **V** knows the vote authentication VA and the plain vote $\mathcal{P}$ as well as the corresponding expected vote verification VV. **V** uses their device **D** to send $(VA, \mathcal{P})$ to the bulletin board **BB**, which forwards it to each $i \in [1, m]$ control component $\mathbf{CC}^{(i)}$. Each $\mathbf{CC}^{(i)}$ responds with the corresponding partial vote verification $vv^{(i)}$, which **BB** forwards to **V**.

V ensures VV' matches VV, else calls the election hotline.

$$
\begin{aligned}
\mathbf{V} \rightarrow \quad & \mathbf{D}: \; VA, \mathcal{P} \\
\mathbf{D} \rightarrow \quad & \mathbf{BB}: \; VA, \mathcal{P} \\
\mathbf{BB} \rightarrow \; & \mathbf{CC}^{(i)}: \; VA, \mathcal{P} \\
\mathbf{CC}^{(i)} \rightarrow \quad & \mathbf{BB}: \; vv^{(i)} \\
\mathbf{BB} \rightarrow \quad & \mathbf{D}: \; \{vv^{(1)}, ..., vv^{(m)}\} \\
\mathbf{D} \rightarrow \quad & \mathbf{V}: \; VV' \leftarrow \oplus_{i=1}^{m} vv^{(i)}
\end{aligned}
$$

Assumptions: OptimalSetupTally, SingleCCHonest, ElectionHotline
Properties: Complete Verifiability, Authentication

Protocol 2: Verifiable internet voting with fewer integrity assumptions.

mation mechanism that CA has been received at the control components.[10] Similar as with the VA, we again provide the bulletin board with a list of hashes representing valid VA.

If the voter did not receive the correct vote verification, they simply abort. As the CA is not published, their vote will simply not be tallied. If the voter still wants to participate in the election, they need to resort to a different voting channel.

In the setup phase, each control component i chooses for each CA a uniform at random partial confirm verification $cv^{(i)}$. During the voting phase, they publish $cv^{(i)}$ when the corresponding CA appears on the bulletin board.[11]

The voter then sums up[12] all the partial confirm verifications $CV' = \oplus_{i=1}^{m} cv^{(i)}$, and checks whether $CV' = CV$ for CV the actual expected confirm verification. If the check succeeds, the voter can be reasonably sure the vote has been confirmed by all control components. If the check fails, the voter calls the election hotline to complain.

[10] This prevents an attacker invalidating a specific vote by dropping or changing CA.
[11] Note that $cv^{(i)}$ are independent of the submitted vote.
[12] Using modular addition, according to the group the cv were picked from.

The voter **V** knows the vote authentication VA and the plain vote $\mathcal{P}$ as well as the corresponding expected vote verification VV. **V** uses their device **D** to send $(VA, \mathcal{P})$ to the bulletin board **BB**, which forwards it to each $i \in [1, m]$ control component $\mathbf{CC}^{(i)}$. Each $\mathbf{CC}^{(i)}$ responds with the corresponding partial vote verification $vv^{(i)}$, which **BB** forwards to **V**.

Further, **V** knows the confirm authentication CA and the expected confirm verification CV. If VV' matches VV, then **V** uses their device **D** to send CA to the bulletin board **BB**, which forwards it to each $i \in [1, m]$ control component $\mathbf{CC}^{(i)}$. Each $\mathbf{CC}^{(i)}$ responds with the corresponding partial confirm verification $cv^{(i)}$, which **BB** forwards to **V**.

V ensures CV' matches CV, else calls the election hotline.

First round-trip to cast vote.		**Second round-trip to confirm vote.**	
$\mathbf{V} \rightarrow$	$\mathbf{D}$: $VA, \mathcal{P}$	$\mathbf{V} \rightarrow$	$\mathbf{D}$: CA
$\mathbf{D} \rightarrow$	$\mathbf{BB}$: $VA, \mathcal{P}$	$\mathbf{D} \rightarrow$	$\mathbf{BB}$: CA
$\mathbf{BB} \rightarrow$	$\mathbf{CC}^{(i)}$: $VA, \mathcal{P}$	$\mathbf{BB} \rightarrow$	$\mathbf{CC}^{(i)}$: CA
$\mathbf{CC}^{(i)} \rightarrow$	$\mathbf{BB}$: $vv^{(i)}$	$\mathbf{CC}^{(i)} \rightarrow$	$\mathbf{BB}$: $cv^{(i)}$
$\mathbf{BB} \rightarrow$	$\mathbf{D}$: $\{vv^{(1)}, ..., vv^{(m)}\}$	$\mathbf{BB} \rightarrow$	$\mathbf{D}$: $\{cv^{(1)}, ..., cv^{(m)}\}$
$\mathbf{D} \rightarrow$	$\mathbf{V}$: $VV' \leftarrow \oplus_{i=1}^{m} vv^{(i)}$	$\mathbf{D} \rightarrow$	$\mathbf{V}$: $CV' \leftarrow \oplus_{i=1}^{m} cv^{(i)}$

Assumptions: OptimalSetupTally, SingleCCHonest, ElectionHotline
Properties: Complete Verifiability, Authentication

Protocol 3: Verifiable internet voting with fewer integrity assumptions and voter feedback.

6.2 Establish privacy

Now we establish privacy. We aim to find a simple protocol which ensures vote secrecy, hence does not reveal the plain vote belonging to a specific voter. Further we aim for fairness, hence no conclusions about the voting result before the tally are possible.

We target again only the voting phase, hence keep the OptimalSetupTally assumption.

6.2.1 Privacy-preserving internet voting

Vote secrecy can be achieved in two ways. We can choose to encrypt the vote, with the encryption done before the vote is seen by untrusted roles or channels, and undone only once the link to the submitting voter has been

separated. Alternatively, we could avoid that the (plain or encrypted) vote is ever being associated to the voter, for example using anonymous channels.

For fairness, we require that no plain vote is seen by untrusted roles or channels before the voting phase ends. We again have two choices: Either the votes are encrypted until the tally phase, or the plain votes are transferred only over confidential channels and to trusted parties. The latter is clearly a too strong assumption, hence we choose instead to encrypt the votes. We reuse this encryption to achieve vote secrecy at the same time.

As we cannot rely on a trusted technical aid, the voter has to be able to encrypt the vote on their own. This can be achieved with a lookup: For each possible voting option, the voter is given its respective encrypted representation. As the voter necessarily has to enter this encrypted representation somewhere to submit the vote, for usability we require this representation to be short.

We implement this by first mapping each plain vote $\mathcal{P}$ to a code vote $\mathcal{C} \in \mathbb{C}$ using the lookup *PtC*. The codes can have a very tight representation, as there only need to be as many codes as there are voting options.

Given the $\mathcal{C}$, we then apply a voter-specific permutation P for encryption. The permutation is only revealed after the vote has been separated from the voter (vote secrecy), and the voting period has ended (fairness). We ensure multiple authorities are involved by letting each define its own permutation, and then chaining all permutations together.[13]

Concretely, we let each control component choose for each voter a uniform at random permutation $p^{(i)} : \mathbb{C} \to \mathbb{C}$. During the setup, these are combined into the chained permutation $P \leftarrow \prod_{i=1}^{m} p^{(i)}$. The chained permutation is applied to the *PtC* lookup to get a voter-specific lookup $PtC_{Id} \leftarrow PtC * P$. The voter receives this PtC_{Id} on their voting sheet so they can map their plain vote $\mathcal{P}$ into the voter-specific permuted code vote $\mathcal{C}$.

As the permutation is different for each voter, to again map $\mathcal{C}$ to its corresponding plain vote and tally the result, the control components need an *Id* associated to the specific code vote. We ensure in the setup phase this *Id* is not attributable to a specific voter.[14]

We require the already defined *SingleCCHonest* assumption: As long as a single CC is honest, it will keep its permutation private long enough for the protocol to achieve vote secrecy and fairness. Further, we require the *NoMetadata* assumption, hence only data explicitly defined in the cryptographic protocol is exchanged. This effectively assumes anonymous channels, an assumption we get rid off in our next iteration.

[13] To recover this chained permutation at a later point, we require each and every authority to reveal its own permutation again.

[14] We omit describing this here, as this is included in the OptimalSetupTally assumption.

Under these trust assumptions, with the proposed mechanism, we already reach all required privacy properties. Vote secrecy is achieved by Id not being attributable to a specific voter. Further, fairness is given as the attacker cannot decrypt the submitted votes.

The voter **V** knows the code vote $\mathcal{C}$ and the id Id. **V** uses their device **D** to send $(\mathcal{C}, Id)$ to the bulletin board **BB**.

$$\begin{aligned} \mathbf{V} &\rightarrow \mathbf{D}: \ \mathcal{C}, Id \\ \mathbf{D} &\rightarrow \mathbf{BB}: \ \mathcal{C}, Id \end{aligned}$$

Assumptions: OptimalSetupTally, NoMetadata
Properties: Vote secrecy, Fairness

Protocol 4: Privacy-preserving internet voting.

6.2.2 Fewer privacy assumptions

Unfortunately, the NoMetadata assumption is unrealistic. The device of the voter is likely registered in the voter's name, or the voter is even logged in with a personal account. The request to the bulletin board is delivered over a network, with the service provider again being able to attribute its origin. The IP, browser fingerprinting, timing or further side channels may enable attribution by other third parties, too.

To ensure the vote stays indeed separated from an identifiable voter, we might require voters to use an anonymous channel when communicating with the bulletin board. Anonymous channels available today include VPNs, mix networks, the onion network, and others. However, these technologies themselves entail trust assumptions. The trust assumption may notably include trust in the device, something we specifically aim to avoid. Further, the selection and use of an appropriate technology may require expert knowledge and make vote submission more demanding for the voter.

Alternatively, we can separate the votes from the voters server-side using a shuffle before decryption. A shuffle takes as input a list of ciphertexts, and outputs a different list of equally many ciphertexts. If executed honestly, the output elements are unlinkable to the input elements, hence keeping the permutation of the values secret. The shuffle is verifiable if a proof is provided that the same cleartext values are represented by the input and output elements.[15] To achieve these properties, verifiable shuffles require input in a specific format.

[15] A dishonest shuffle could instead drop all but one vote which would then be decrypted.

We cannot assume the voter is able to calculate their ready-to-shuffle vote by hand, and we cannot use a technical aid (as this again would require privacy assumptions at said technical aid). As before, we could instead use a lookup, with the voter mapping their plain vote to their ready-to-shuffle vote. However, usability is low compared to the code votes approach, as the ready-to-shuffle votes are long ciphertexts.

Instead, we introduce another mapping ***CtE*** in the tally phase, which maps for each voter the short code votes to their respective ready-to-shuffle representation. This mapping can be done in a verifiable way, hence no privacy (nor verifiability) properties are in danger, but we enable the voter to continue to use the much simpler code votes. Crucially, the plain text of the ready-to-shuffle representation corresponds to the actual plain vote, hence applying the voter-specific mapping $CtE_{Id} \in \boldsymbol{CtE}$ reverses the voter-specific permutation.[16]

With this idea, we can now remove the NoMetadata assumption, while our properties remain unchanged. For vote secrecy, the reasoning changes, as it is now guaranteed by the privacy-preserving properties of the verifiable shuffle done in the tally phase. Fairness is still preserved as the votes remain encrypted.

The voter **V** knows the code vote $\mathcal{C}$ and the id Id. **V** uses their device **D** to send $(\mathcal{C}, Id)$ to the bulletin board **BB**.

In the tally phase, the votes are verifiably shuffled and decrypted.

$$\mathbf{V} \rightarrow \mathbf{D}\colon\ \mathcal{C}, Id$$
$$\mathbf{D} \rightarrow \mathbf{BB}\colon\ \mathcal{C}, Id$$

Assumptions: OptimalSetupTally
Properties: Vote secrecy, Fairness

Protocol 5: Privacy-preserving internet voting with fewer trust assumptions.

6.3 Combine proposals and establish setup and tally

We now combine the verifiable and privacy-preserving voting phase proposals. Then we define an appropriate setup and tally phase.

[16]Formally: For a voter Id submitting $\mathcal{C} = PtC_{Id} * \mathcal{P}$, the corresponding ready-to-shuffle representation $\mathcal{E} = CtE_{Id} * \mathcal{C}$ represents $\mathcal{P} = \mathcal{C} * PtC_{Id}^{-1}$.

6.3.1 Combine proposals

We combine the two proposals by merging the privacy-preserving protocol into the first round-trip of the verifiable protocol. Concretely, we replace the plain vote $\mathcal{P}$ with the code vote $\mathcal{C}$, and add the *Id* to the voter requests[17]. As the voter now submits $\mathcal{C}$ instead of $\mathcal{P}$, a necessary adaptation is also to choose the vote verifications *VV* (respectively the partial vote verifications *vv*) in relation to $\mathcal{C}$ instead of $\mathcal{P}$.

With a combination of all assumptions of the two originating proposals, we argue to also reach a combination of all their properties. The mechanism for individual verifiability is preserved as the *VV* still validate the correct vote has been received. Universal verifiability and authentication are unchanged. Vote secrecy and fairness is preserved when *VV* does not leak the plain vote, and unaffected by the second round-trip, as the second round-trip is independent of a specific vote.

We argue that our optimality target is preserved by this combination. First, we note that the combined proposal does not have more round-trips than the verifiability-only proposal.[18] Second, we observe that while each request has multiple values, each value has a single purpose. We could instead opt for a single value per request (and join the purposes together), but we believe this is more complicated to understand, less efficient to operate[19] and leads to worse usability[20].

[17] Note that we could choose to drop *Id* as *VA* provides strictly stronger properties: *VA* is both unique *and* hard-to-guess. We however keep the *Id* as it is useful to defend against DDoS at the bulletin board.

[18] Reducing the number of round-trips seems impossible, no matter the mechanism.

[19] As argued before, with a stand-alone *Id* the bulletin board can be defended better against DDoS.

[20] Consider the case where multiple $\mathcal{C}$ can be cast at the same time. With the long *VA* separated from the short $\mathcal{C}$, *VA* can be reused for all $\mathcal{C}$, reducing the amount of data which needs to be entered by the voter.

The voter **V** knows the vote authentication VA, the id Id, the plain vote $\mathcal{P}$ - using the voter-specific code vote lookup PtC_{Id} - mapped to the code vote $\mathcal{C}$, as well as the corresponding expected vote verification VV. **V** uses their device **D** to send $(Id, VA, \mathcal{C})$ to the bulletin board **BB**, which forwards it to each $i \in [1, m]$ control component $\mathbf{CC}^{(i)}$. Each $\mathbf{CC}^{(i)}$ responds with the corresponding partial vote verification $vv^{(i)}$, which **BB** forwards to **V**.

Further, **V** knows the confirm authentication CA and the expected confirm verification CV. If VV' matches VV, then **V** uses their device **D** to send CA to the bulletin board **BB**, which forwards it to each $i \in [1, m]$ control component $\mathbf{CC}^{(i)}$. Each $\mathbf{CC}^{(i)}$ responds with the corresponding partial confirm verification $cv^{(i)}$, which **BB** forwards to **V**.

V ensures CV' matches CV, else calls the election hotline.

In the tally phase, the votes are verifiably shuffled and decrypted.

First round-trip to cast vote.	**Second round-trip to confirm vote.**
$\mathbf{V} \rightarrow \mathbf{D}$: $Id, VA, \mathcal{C}$	$\mathbf{V} \rightarrow \mathbf{D}$: Id, CA
$\mathbf{D} \rightarrow \mathbf{BB}$: $Id, VA, \mathcal{C}$	$\mathbf{D} \rightarrow \mathbf{BB}$: Id, CA
$\mathbf{BB} \rightarrow \mathbf{CC}^{(i)}$: $Id, VA, \mathcal{C}$	$\mathbf{BB} \rightarrow \mathbf{CC}^{(i)}$: Id, CA
$\mathbf{CC}^{(i)} \rightarrow \mathbf{BB}$: $vv^{(i)}$	$\mathbf{CC}^{(i)} \rightarrow \mathbf{BB}$: $cv^{(i)}$
$\mathbf{BB} \rightarrow \mathbf{D}$: $\{vv^{(1)}, ..., vv^{(m)}\}$	$\mathbf{BB} \rightarrow \mathbf{D}$: $\{cv^{(1)}, ..., cv^{(m)}\}$
$\mathbf{D} \rightarrow \mathbf{V}$: $VV' \leftarrow \oplus_{i=1}^{m} vv^{(i)}$	$\mathbf{D} \rightarrow \mathbf{V}$: $CV' \leftarrow \oplus_{i=1}^{m} cv^{(i)}$

Assumptions: OptimalSetupTally, SingleCCHonest, ElectionHotline
Properties: Complete Verifiability, Authentication, Vote secrecy, Fairness

Protocol 6: Verifiable and privacy-preserving internet voting.

6.3.2 Define Tally

The tally has to be performed in a verified and privacy-preserving way without relying on any trust in authorities except that a single control component is honest. Given the voting phase, the tally almost directly follows.

To tally, the confirmed votes are chosen and mapped to their ready-to-shuffle representation using ***CtE***. Afterwards, one component after the other applies their verified decryption-shuffle to the previous component's output until the last control component results with the plain votes.[21] The inputs and proofs of the shuffles are verified by the other components. If all attest correct execution, the plain votes are counted and the result is announced.

[21] Note that the voter-specific ready-to-shuffle representation already represents the true plain vote (see section 6.2.2).

To produce the mapping $CtE_{Id} \in \boldsymbol{CtE}$ which maps a vote $\mathcal{C}$ submitted under Id to its ready-to-shuffle representation $\mathcal{E}$, we necessarily need to combine each control component's partial permutation $p^{(i)}$ into the chained permutation $P \leftarrow \prod_{i=1}^{m} p^{(i)}$. However, in the tally phase, the Id has to be assumed attributable to a voter.[22] Therefore, executing the product and binding it to the respective $\mathcal{C}$ must be done in an obfuscated and verifiable manner. There is no obvious solution how this can be done, but it will likely involve rather complex cryptography we explicitly try to avoid.

As a simple alternative, we instead produce ***CtE*** during the setup phase and commit it to the bulletin board. From a security analysis perspective, producing and delivering CtE_{Id} is no different than producing and delivering the voter-specific PtC_{Id}.

The SingleCCHonest assumption is still enough to perform the verified tally: Each CC checks the inputs and proofs of the shuffle of all other CCs, and draws attention to failing proofs. Both privacy and universal verifiability guarantees remain preserved.[23] We replace the OptimalSetupTally assumption with the *OptimalSetup* assumption for obvious reasons.

The bulletin board **BB** stores confirmed votes of the form $(Id, \mathcal{C})$. Using ***CtE***, the first control component $\mathbf{CC}^{(1)}$ maps each $\mathcal{C}$ to their ready-to-shuffle equivalent and then performs the first shuffle. Each control component $\mathbf{CC}^{(i)}$ for $i \in [2, m]$ performs their own shuffle on the output of the previous. $\mathbf{CC}^{(m)}$ outputs the plain votes. If no **CC** complains about the shuffles and decryptions of others, the plain votes are counted and the result is announced.

Assumptions: OptimalSetup, SingleCCHonest
Properties: Vote secrecy, Fairness, Universal Verifiability

Protocol 7: Tally phase of a verified and privacy-preserving internet voting.

6.3.3 Define Setup

With all other parts of the protocol set, the setup could be defined straightforwardly: A trusted setup authority receives as input the data from the election authorities (like eligible voters, voting options and other election configuration), prepares the randomized key material, and then sends it to the other involved parties.

However, this would prevent us from effectively auditing the trusted authorities, something we would like to do to reduce the associated risk. If the auditors are supposed to effectively check for honest execution, the involved

[22] As we dropped the NoMetadata assumption.
[23] Authentication and individual verifiability are not a concern during the tally phase.

algorithms cannot involve non-deterministic elements, as auditors cannot be expected to distinguish honest randomization from phony randomization. The latter however potentially gives the attacker full knowledge about the output of the algorithms, without even needing an active communication channel. We resolve this by letting the trusted authorities only execute deterministic algorithms, and move randomization to the control components.

Further, we want to ensure the trusted authorities cannot leak unauthorized data. We ensure this indeed can be observed by simplifying our communication model. In our protocol, we restrict communication towards untrusted components to unencrypted data. Further, the trusted authorities can be fully wiped after after operations, which is before the voting phase starts.

We additionally consider that the auditors themselves are not fully trusted, with the strongest trust assumption we can realistically (and lawfully) take being *SingleAuditorHonest*. Simply giving the auditors full access to the trusted authorities' state would then break whatever the reason is these authorities need to be trusted. We instead implement audits using a probabilistic approach: Each auditor insert dummy voters, then verifies the execution is honest for these dummy voters.

With the restricted functionality and communication, and the probabilistic audits, we do effectively harden our defence of the trusted authorities against attackers. Attackers that were only active in the past, but no longer have access to the system, can not learn anything about the upcoming election. Further, currently active attackers are not able to export their knowledge unnoticed out of the system.

Given the restrictions, we design the setup (detailed in section 7.2.1) as follows: The setup authority publishes the number of voters and the encoding of plain votes on the bulletin board. The control components then generate appropriate secret cryptographic material, and send it in encrypted form to the setup authority. The setup authority decrypts and combines the key material. It sends the bulletin board the hash of the authentication secrets, as well as the values required to execute the shuffle in the tally phase. Further, it prepares the to-be-printed material for the voters, and sends this to the printer authority.[24]

We reuse the SingleCCHonest assumption, and ensure the combination of key material requires only a single honest control component to be effective. Further, we introduce the *SetupHonest* and *PrinterHonest* assumptions, as both see the secret key material delivered to the voter. For the authorities involved in the setup to communicate securely, we assume an established PKI, and name this assumption *SetupPKI*.

[24]It aligns with the organisational realities (and the legal requirements) that the printer authority is a separate instance.

6.4 Summary

Based on the setting as seen in chapter 5, we iteratively constructed an internet voting protocol. For each step, we argued informally under which assumptions the verifiability and privacy properties were established and preserved.

First, we defined a voting phase fulfilling the verifiability properties. In the first round-trip, the voter sends the vote together with a vote authentication to the bulletin board and expects back a vote-specific verification. If the verification matches to what the voter expects, the voter initiates a second round-trip by sending their confirm authentication. If the voter again receives back the expected confirm verification, the vote was cast and confirmed successfully.

Then, we defined a privacy-preserving voting phase. The voter is instructed to send an encryption of their vote to the server. We observe that we cannot assume that the voter has used an anonymous channel, hence before decryption, we have to verifiably shuffle the received votes. To improve usability, we introduce a lookup of short representations of an encrypted vote (which the voter submits) to the long ciphertexts required for the shuffle.

Finally, we combine the two proposals together, arguing that the combination is still as simple as possible in the given setting. We define an appropriate tally phase, and sketch the most important aspects of the setup phase.

Chapter 7

Formal Proposal

Based on the setting established in chapter 5, and the draft established by the iterative proposal in chapter 6, we now finally propose our protocol formally.

In its full detail, this would take dozens of pages, while the core ideas would still be hard to grasp. We choose a different approach: We describe the *Core Protocol*, which contains all mechanisms specific to internet voting, in full detail. This core protocol then assumes an established *Infrastructure* with standard cryptographic primitives. We will clearly describe how the core protocol interacts with the infrastructure (and formulate precise requirements) but provide only syntax (and no algorithms) of the latter.

We first repeat the setting we are in. Then, we define the infrastructure we depend on, and go on with describing the protocol in its full detail. At the end of this chapter, we place an informal security analysis, and propose some extensions to the protocol.

Model We use the same roles as established in chapter 5. The setup authority prepares the election. The printing authority receives data from the setup authority to forward to the voter over the postal channel. The voter casts and confirms their vote. Throughout the protocol, control components together generate key material, register cast and confirmed votes, and tally in the end. All communication (except over secure channels) is stored on the publicly-reachable append-only bulletin board. The protocol execution is further observed by auditors, which may detect when trust assumptions are broken.

The roles are connected to each other over bidirectional untrusted channels, except the auditors (which may not send any messages at all), and the printer authority (which may only send the voter a postal letter over the assumed secure postal channel).

We assume a covert adversary, which may deviate arbitrarily from the protocol, but does not wish to be "caught" doing so [12]. This attacker model notably excludes many disruption of service attacks (like components refusing to process input, or flooding by attributable requests).

Our strongest trust assumption is that we have to rely on a fully trusted[1] setup and printer authority. We simplify observation and thereby reduce risk of this trust assumption by restricting outward communication (only in plain or towards individual voters) and choosing a simplified execution model (concretely, no randomness is produced at these trusted authorities[2]). Further, both authorities are not needed any more after the setup phase, hence their data can be fully wiped before the election starts.

During the whole course of the protocol, we rely on a single honest control component and a single honest auditor (while multiple control components and auditors are engaged). It is up to the operator how at least one honest role is realistically guaranteed. For example, each control component could be hosted by a different canton, with each using their own implementation of the protocol. Trustworthy auditors could be most likely found in the often already established voting commissions on municipal level.

Further, we rely on the trust assumptions our infrastructure requires. The infrastructure has to be chosen in such a way that it does not weaken our model, which in our opinion is realistic.

Importantly, we place no assumptions in the voting device, contrary to other proposals for vote électronique.

Properties We achieve all properties as required and defined by law. This concretely includes individual verifiability, universal verifiability, vote secrecy, fairness and authentication. We formally define the properties in chapter 8, and then proof the protocol implements them in chapter 9.

We reach a notion of accountability through the bulletin board, which stores the progress of the protocol with each step being attributable. We further improve the system towards availability, usability and transparency.

7.1 Infrastructure

Before describing the core protocol, we detail the infrastructure our protocol relies upon.

We avoid defining the infrastructure algorithms in full detail, to reduce complexity of this document and of the protocol. We however fully specify necessary "glue" to plug-and-play an appropriate cryptographic primitive. At

[1] We refer to an authority as trusted when it operates correctly and does not leak secrets.
[2] Randomness predictable to the attacker may emulates outwards communication.

the same time, the mechanisms are cleanly abstracted away from the protocol, which allows the infrastructure to evolve on its own. Progress in research or industry on the infrastructure level can therefore continuously be integrated into the system without adapting the core protocol.

The mechanisms have mostly industry-standard solutions. For the verifiable shuffle, we propose to use Bayer-Groth as used by Swiss Post [235], or Wikström as used by CHVote [139].

The infrastructure and the core protocol rely on cryptographic parameters such as group sizes. We assume these parameters are picked appropriately to reach the aimed for security level, and all participants of the protocol (and the infrastructure) have access to these values.

7.1.1 Channels

Besides the untrusted channels, we rely on authentic and secure channels for some parts of the protocol. Concretely, to implement the bulletin board, we require authentic channels from and to the bulletin board. In the setup phase of the protocol, we additionally require some secure channels.

Authentic channels We require bidirectional authentic channels from the bulletin board which connect to the setup authority, the printer authority and each control component. This concretely means whenever a message is sent from or to the bulletin board, the sender is undeniably identified and therefore attributable. This also holds for third parties; hence the bulletin board can convince the control component when some message originated at the setup authority.

We specifically note that the protocol may be run multiple times in parallel, a reasonable assumption as different eligibilities and voting cycles each require their own instantiation of the protocol. The authenticated communication must therefore provide something similar as injective agreement as defined by Lowe [181].

We use $\xleftarrow{A}$ and $\xrightarrow{A}$ to denote communication sent over the authenticated channel. We describe in section 7.1.2 how this is denoted exactly.

Secure channels In the setup phase of the protocol we require some secure channels, hence channels which are both authentic as well as secret. Concretely, we require secure channels from each control component to the setup authority, and another secure channel from the setup authority to the printer authority.

One complication is that the setup authority (per our model) should not produce randomness, which is typically required from such a mechanism.[3] The encryption randomness could instead be produced by the control components, like the voter key material, with the printer authority validating the expected randomness was used.[4]

We use $\xleftarrow{s}$ and $\xrightarrow{s}$ to denote communication sent over the secure channel.

7.1.2 Building Blocks

Besides the channels, we require some more complex primitives we detail here. The bulletin board is our append-only storage, which makes sure the progress of the protocol is traceable. The verifiable shuffle ensures the votes are separated from their voters before the votes are decrypted. The election hotline is our way out to resolve conflicts not possible within the cryptographic protocol.

Hash We need a collision resistant hash function (like SHA-256) supporting input of arbitrary size. We denote this hash function as Hash.

Bulletin Board The bulletin board is an append-only storage. It publishes a new entry for each received authentic message and each result of own computations. Each such published entry further references its previous entry, to enforce an explicit ordering and make detection of missing messages easier. Besides making all entries publicly accessible, entries are also published over the authentic channels towards all control components.

We specify that the control components must only react to the entries published over the authentic channel. Each entry published is then verified by the control components whether it adheres to the specification. As at least one control component is honest, this avoids us placing any trust assumptions for correct execution in the bulletin board: Any misbehaviour[5] is now detectable and therefore outside of our adversary model. Note that in the setup phase the control components ensure they all have the same initial view, and in the tally phase, the control components ensure they all have the same view of the confirmed votes.[6]

We further specify that the bulletin board only reacts to entries received over its authentic channels (besides the requests of the voters, which each have a single-use authentication secret attached). This prevents malicious authorities from flooding the bulletin board with requests (as the requests

[3] Secure channels need encryption, and IND-CPA security or stronger needs randomness.
[4] This needs to be carefully designed to not expose the randomness to the attacker.
[5] Executing computations wrongly, or attempting to removing or change entries later on.
[6] Different views are clearly an attack and therefore again outside the attacker model.

are attributable, this scenario is again outside the attacker model), and external authorities from having any request processed (as they have no valid authentication whatsoever).

This design puts the bulletin board, the only publicly reachable service in our protocol, in a strong position to defend against DDoS attacks.[7] It only needs to process authenticated messages, and detecting unauthenticated requests is usually rather efficient.

We omit in our protocol explicitly describing whenever the bulletin board sends messages, as it is obvious: Whenever a new message is received, or a computation is executed, the bulletin board publishes a new corresponding entry. The control component receive this entry over the authenticated channel, and react to it if appropriate. Other authorities are expected to pull the bulletin board on demand.

We however explicitly describe whenever the bulletin board receives messages, using our authentic channel notation.

Verifiable shuffle When a voter sends a request to the (untrusted) bulletin board, we have to assume the bulletin board associates the request with an identifiable voter.[8]

The verifiable shuffle ensures the vote and the voter are separated before the vote is decrypted. To ensure the decryption cannot be done over unshuffled votes, we require a decryption scheme which needs involvement of all control components (as the single honest control component we assume will refuse decryption of incorrectly or not shuffled votes). As the shuffling and the decryption might be done by the same cryptographic primitive, we define our syntax to do both tasks at the same time.

We define the verified decryption-shuffle as consisting out of the following algorithms:

- $\mathsf{SKGen}()$ is a randomized function which returns a key pair $(sk_S, pk_S) \in (SK, PK)$, for sk_S the secret key and pk_S the public key.
- $\mathsf{SKAgg}(\boldsymbol{pk}_S)$ is a deterministic function which takes a list of public keys $pk_S \in \boldsymbol{pk}_S$ such that $\forall pk_S \in PK$ and outputs an aggregated public key $pk_S \in PK$.
- $\mathsf{SEnc}(r, pk_S, m)$ is a deterministic function randomized by $r \in R \subseteq \{0,1\}^*$ which takes a public key $pk_S \in PK$ and a plaintext $m \in M \subseteq \{0,1\}^*$ and outputs a ciphertext $c \in C \subseteq \{0,1\}^*$.

[7] Note that operating a public facing service is expensive and requires highly specialized appropriate defences like DDoS protection. Having only a single exposed component reduces cost. Further, as the component is untrusted, operators can be freely chosen.

[8] See the discussion in section 6.2.2 why this is the case, and why voters cannot be assumed to use an anonymous channel.

- $\mathsf{SShuffleDecrypt}(\mathbf{c}, sk_S)$ is a randomized function which takes a list of ciphertext $\mathbf{c}$ such that $\forall c' \in C$, and a secret key $sk_S \in SK$ as arguments. It outputs a list of shuffled ciphertexts $\mathbf{c}'$ such that $\forall c \in C$, and a proof $\pi \in \Pi$. If the function has been called once for every secret key of which their aggregated public key was used in the encryption, with its ciphertext list input being the ciphertext list output of the previous invocation, the function instead outputs a plaintext $m \in M$.[9]
- $\mathsf{SVerifyShuffleDecrypt}(\mathbf{c}, \mathbf{c}', \pi)$ is a deterministic function which takes two lists of ciphertext, $\mathbf{c}$ such that $\forall c \in C$, and $\mathbf{c}'$ such that $c' \in C$, and a proof $\pi \in \Pi$. It outputs either true or false.

Definition 7.1 (Terminology) We use terminology as follows:

- A *Valid Public Key* was output by SKGen or was output by SKAgg which arguments were only valid public keys.
- The *Decryption Secret Keys* of a valid public key is the secret key output together with the public key by SKGen. For aggregated public keys, the term refers to the list of the respective secret keys.
- A *Valid Ciphertext* is a ciphertext output by SEnc, for any $r \in R$, message $m \in M$ and valid public key.

Definition 7.2 (Correctness) Given the terminology of definition 7.1, we define correctness as follows:

Given any list of valid ciphertexts $\mathbf{c}$ - representing a list of plaintext $\mathbf{m}$, encrypted using the valid public key pk_S - when:

- calling SShuffleDecrypt for each decryption secret key sk_S of pk_S;
- given as input the ciphertext list of the previous invocation (the first invocation gets directly $\mathbf{c}$) and the current sk_S;

results a list of ciphertext $\mathbf{c}'$ and a proof $\pi \in \Pi$ for which

- $\mathbf{c}'$ represents the same plaintext values as $\mathbf{c}$, but in a randomly permuted order;
- SVerifyShuffleDecrypt when passed $\mathbf{c}$, $\mathbf{c}'$ and π returns true.

Both constraints also hold for when SShuffleDecrypt is called with the last decryption secret key and therefore outputs $\mathbf{m}'$.

[9] We abuse notation like this to avoid defining an Encode and Decode function from the ciphertext space to the plaintext space and back.

SVerifyShuffleDecrypt will return false if π does not proof that $\mathbf{c}$, $\mathbf{c}'$ contain the same plaintext values.[10]

Security assumptions are made explicit in chapter 9. We however highlight some critical details of how the shuffle is embedded into our protocol:

- Each control component executes SKGen, which leads to at least one key pair using proper randomness.
- The setup authority executes functions SKAgg and SEnc. We note that these are either deterministic or their randomness is explicitly passed into the function.
- Each control component only executes SShuffleDecrypt if for all previous invocations SVerifyShuffleDecrypt returns true (which prevents the honest control component from decrypting some incorrectly shuffled list of ciphertext).
- Each control component will execute SShuffleDecrypt once, which ensures at least once the permutation is done randomly.
- Each control component checks the proofs of all invocations after its own SShuffleDecrypt call.

All operations and values which belong to the shuffle are drawn upon yellow background, as this makes it easier to recognise the syntax.

Election hotline The election hotline is where the voter can complain if their vote has been cast, but not confirmed correctly. The election hotline is then able to invalidate the vote, as foreseen by law [52, appendix 4.9]. It is not part of the protocol how the claim is authenticated, nor how the vote is then unvalidated.

To reduce the load on the election hotline, we have introduced separate round-trips for casting and confirming a vote (see section 6.1.3). The election hotline is therefore only called when the attacker did not interfere when casting the vote but interfered when confirming it, and we assume this scenario to be unlikely.[11]

We do not establish syntax for the election hotline, rather simply refer to it.

[10] Note that the function is not required to verify whether the permutation was applied randomly or not, as this is typically not possible.

[11] We might further reduce potential load on the election hotline by introducing additional round-trips (e.g. to "finalize" a confirmed vote). However note that this problem cannot be solved entirely within the cryptographic protocol, per the two generals problem, and we expect the benefits of additional round-trips relative to their cost to be slim.

7.2 Core Protocol

The protocol is divided into three phases. It starts with the setup phase, which ensures all parties receive all required cryptographic material before the election starts. In the voting phase, the voter submits their vote and performs their individual verifiability check. In the final tally phase, the votes are shuffled, decrypted and counted.

To avoid the complexity of introducing all parts of the protocol at the same time, we structure our presentation as follows. In this section, we focus on the values exchanged between the roles, but give only an intuition of what these values are composed of. In the next section, we explicitly define the values themselves and present pseudo-code for all involved algorithms which produce or process them.

Roles As in our setting, auditors are not active participants of the protocol. One might however employ auditors to give additional assurances that the trust assumptions are not broken, which we describe informally in appendix C.

Notation We use boldface for an unordered list of elements, for example $\mathbf{a} = [a_1, a_2, ...]$.

Expression	Example	Result
get range	$\mathbf{L} \leftarrow [1,4]$	$\mathbf{L} = [1,2,3,4]$
get size	$N_{\mathbf{L}} = \lvert\mathbf{L}\rvert$	$N_{\mathbf{L}} = 4$
choose random	$x \xleftarrow{r} \mathbb{N}$	$x = 2$
choose match	$x \xleftarrow{.} (2,.) \in \{(1,a),(2,b)\}$	$x = b$

We denote the combination of permutations using $*$, like $[0 \to 1, 1 \to 0] * [0 \to 0, 1 \to 1] = [0 \to 1, 1 \to 0]$. We overload the $*$ operation to also query lookups[12], like $a * [a \to 2] = 2$. We may also revert lookups, like $2 * [a \to 2]^{-1} = a$.

We denote modular addition with $\oplus$. The modulo is defined over the group from which the individual values have been drawn from (for example if $x, y \xleftarrow{r} \mathbb{Z}_v$ and $\mathbf{L} = [x, y]$, then $\oplus\mathbf{L} = \sum \mathbf{L} \bmod v$).

For partial values of the control components we use lowercase with an index top right (for example, $vv^{(i)}$ for the partial vote verification of $CC^{(i)}$). When the partial values have been combined, we use the uppercase equivalent (for example, VV for the vote verification).

When a party encounters an *assert* with an invalid expression (like *assert false*), then the party aborts processing the particular message.

[12] We refer to lookups for permutations where the source and target groups do not match.

7.2.1 Setup

The election authorities prepare for each election the following values:

Description	Notation
Voters (includes postal addresses)	$\mathbb{V}$
The first voter id	Id_v
Codes	$\mathbb{C}$
Plain to Codes lookup	PtC

$\mathbb{V}$ includes all information relevant to send the voting material to the voter. Only its size $|\mathbb{V}|$ will be published on the bulletin board.

Id_1 has to be chosen in such a way that the range $[Id_1, Id_1 + |\mathbb{V}|]$ produces no overlaps with any other active election.

$\mathbb{C}$ is the collection of codes where each represents a voting option. $|\mathbb{C}|$ must be equal the number of available voting options. As the voter has to enter these codes by hand, it is advisable to choose values that are as easy and short to enter as possible. Further, the codes should not represent actual voting options (to not confuse the voter) or indicate preference of some sort (to not influence the voter). For example, the alphabet may be suitable, without ambiguous characters such as *0O* or *1Il*.

PtC maps each voting option to a unique code. It is fine if PtC is predictable, as for each voter it will be randomly permuted into the voter-specific PtC_{Id}. For example $['1.Yes' \rightarrow A, '1.No' \rightarrow B, '1.Abstain' \rightarrow C, '2.Yes' \rightarrow D, ...]$.

Setup phase (1/2) The setup authority establishes key material with the control components (see Protocol 8).

The administrator starts the process by entering $(Id_1, |\mathbb{V}|, PtC)$ into the setup authority. This is sent to the bulletin board, which generates a sufficiently large set of ***Id*** to have one Id per voter.

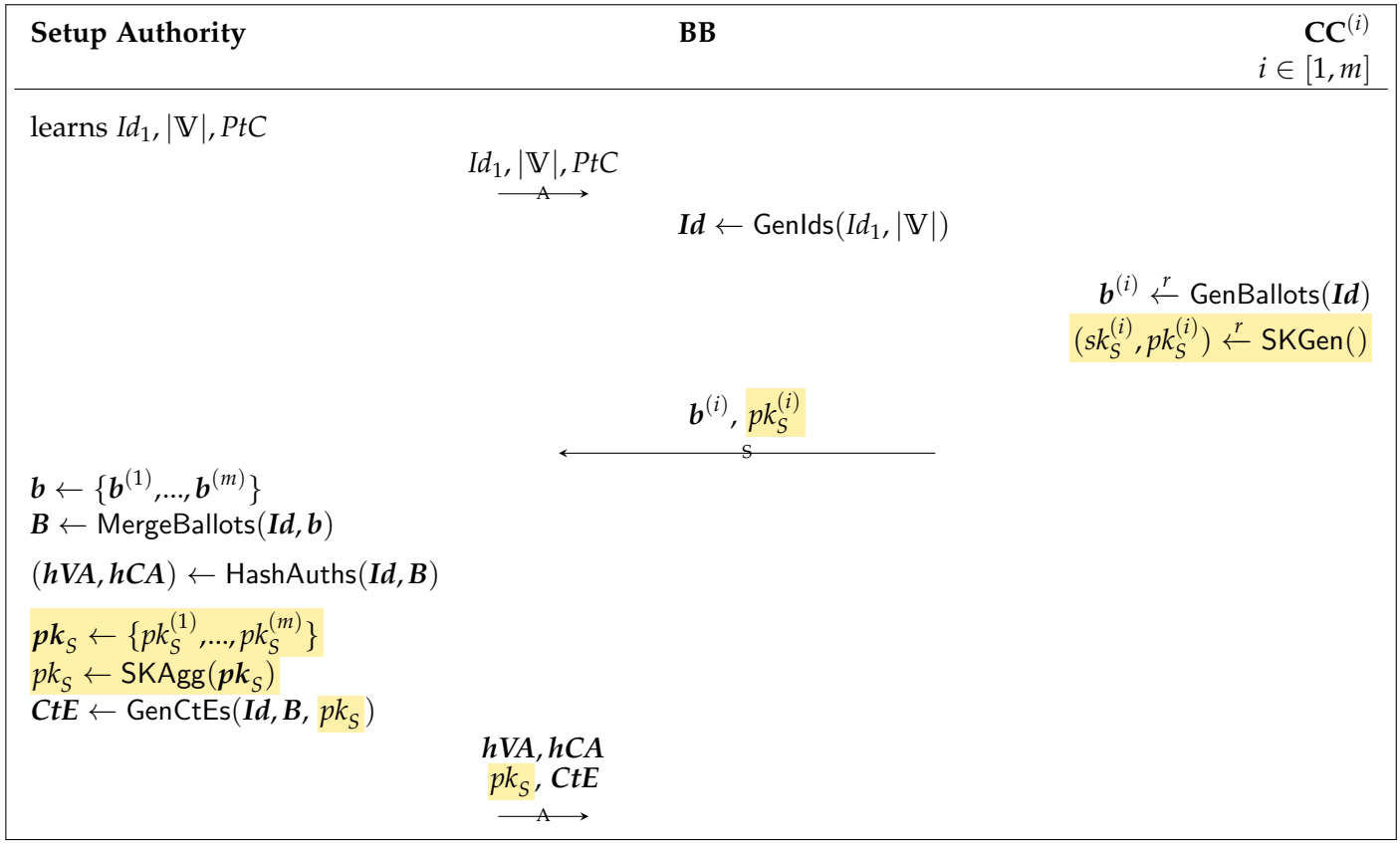
Each control component generates for each voter a partial ballot $b^{(i)}$. For the tally phase, each control component generates a shuffle key pair. The partial ballot of each voter as well as the shuffle public key are sent back to the setup authority over the secure channel.

The setup authority merges each voter's partial ballots into the ballot B. It then publishes each hashed authentication secret hVA and hCA on the bulletin board.[13]

To prepare the shuffle, the setup authority aggregates the shuffle public keys. The resulting aggregated public key pk_S is used to create for each voter a lookup CtE_{Id}, which maps each possible code vote to a ciphertext

[13] For the bulletin board to check whether a voter's request is properly authenticated.

representing the corresponding plain vote. All $\boldsymbol{CtE}$ and the pk_S are posted on the bulletin board.



Protocol 8: Setup phase (1/2) where the setup authority establishes key material with the control components (CC). The bulletin board (BB) facilitates, and receives authentication hashes and key material to store.

Setup phase (2/2) The setup authority delivers the secret material to the voter via the printer authority (see Protocol 9).

The administrator continues the setup by entering $\mathbb{V}$ into the setup authority. Entering $\mathbb{V}$ only in the second phase of the setup helps to preserve privacy: Outgoing channels are from now on only towards individual voters, which is easier supervised than requests to the bulletin board.

The setup authority assigns[14] each voter a ballot B and generates their personal ballot sheet BS out of B. The ballot sheet notably contains all authentication secrets and the voter-specific permutation of the codes. It may

[14] Ballots are voter-independent until now, hence this assignment can be arbitrary.

already be represented in a format which the printer can directly output (for example, a print-optimized PDF).

The ballot sheets are sent to the printer over the second secure channel. The printer prints the ballot sheet and sends it to the corresponding voter over postal mail, another assumed secure channel.

Setup Authority	**Printer**	**Voter**$_i$ $i \in [1, \lvert\mathbb{V}\rvert]$
learns $\mathbb{V}$ knows $\boldsymbol{Id}, \boldsymbol{B}, PtC$		
$\boldsymbol{BS} \leftarrow \mathsf{GenBallotSheets}(\boldsymbol{Id}, \boldsymbol{B}, \mathbb{V}, PtC)$		
$\boldsymbol{BS} \xrightarrow{\text{S}}$		
	$BS_i \xrightarrow{\text{S}}$	

Protocol 9: Setup phase (2/2) where the setup authority prepares the ballot sheets for the printer, and then the printer sends them to the voter.

7.2.2 Voting phase

The voter casts and confirms their vote using the ballot sheet received over postal mail. The bulletin board checks authentication and maintains the list of cast and confirmed votes. The control components observe the actions of the bulletin board, and work together to recover the verification based on their view of the cast and confirmed votes. If the control components' views are consistent, the correct verifications for the voter are recovered, which the voter checks against their ballot sheet.

Voting phase (1/2) The voter casts their vote (see Protocol 10).

When the voter has decided on their preferred voting option $\mathcal{P}$, they use PtC_{Id} to map $\mathcal{P}$ to the corresponding code vote $\mathcal{C}$. Together with their *Id* and the vote authentication *VA* the voter sends $\mathcal{C}$ to the bulletin board.[15]

The bulletin board ensures the authentication and the validity of the vote, otherwise the request is simply ignored. Then the bulletin board checks this is indeed the first vote cast by that voter, otherwise the bulletin board

[15]Technically, the *Id* is not needed, as the hash of *VA* already uniquely identifies a voter. However, it allows to debug failed authentication (has the voter participated in the wrong election?) and might speed up authentication (with the first layer only checking whether the *Id* is valid, before even hashing *VA*).

responds with the earlier cast vote (and the respective partial vote verifications).[16] If all checks succeed, the vote is published and the control components are notified.

Each control component extracts the applicable partial vote verification $vv_{\mathcal{C}}$ and sends it to the bulletin board, which forwards it to the voter.

The voter extracts the expected vote verification VV from the ballot sheet, and continues the voting phase if the combination of the partial vote verifications result in the same value. Otherwise, the voter aborts.[17]

For an election with n selectable voting options, the exchange is done with n different $\mathcal{C}$ at the same time. Partial votes are not allowed (else the attacker might drop parts of the vote), hence all voters must submit exactly n votes.

For an election with only certain combinations of voting options allowed, the $\mathbb{C}$ and its voter-specific permutations are appropriately structured into subgroups. For each subgroup, it is enforced that exactly the appropriate number of options is cast. For example, for an election with a yes/no question and a choose-two-out-of-three-options question, $\mathbb{C}$ would be structured in two subgroups, like $\mathbb{C}_1 = \{A, B\}$ and $\mathbb{C}_2 = \{C, D, E\}$. The bulletin board then enforces that exactly three $\mathcal{C}$ are cast, with exactly one belonging to $\mathbb{C}_1$ and exactly two belonging to $\mathbb{C}_2$.

[16] If the voter has already confirmed their vote, the bulletin board ignores the request.
[17] The voter may still use a different voting channel, if available.

Voter		**BB**		**CC**$^{(i)}$ $i \in [1,m]$
knows BS		knows $\boldsymbol{hVA}$		knows $\boldsymbol{b}^{(i)}$
decides $\mathcal{P}$ $(Id, VA, ..., PtC_{Id}, ...) \leftarrow BS$ $\mathcal{C} \xleftarrow{\cdot} (\mathcal{P}, .) \in PtC_{Id}$				
	$\xrightarrow{Id, VA, \mathcal{C}}$			
		asserts $(Id, \mathsf{Hash}(VA)) \in \boldsymbol{hVA}$ asserts $\mathcal{C} \in \mathbb{C}$ asserts $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Cast}$		
		$\boldsymbol{\mathcal{C}}_{Cast} \leftarrow \boldsymbol{\mathcal{C}}_{Cast} \cup \{(Id, \mathcal{C})\}$		
				$(..., \boldsymbol{vv}^{(i)}, ...) = b_{Id}^{(i)}$
			$\xleftarrow[\text{A}]{vv_{\mathcal{C}}^{(i)}}$	
		$\boldsymbol{vv}_{\mathcal{C}} \leftarrow \{vv_{\mathcal{C}}^{(1)}, ..., vv_{\mathcal{C}}^{(m)}\}$		
	$\xleftarrow{\boldsymbol{vv}_{\mathcal{C}}}$			
$\{vv_{\mathcal{C}}^{(1)}, ..., vv_{\mathcal{C}}^{(m)}\} \leftarrow \boldsymbol{vv}_{\mathcal{C}}$ $VV'_{\mathcal{C}} = \oplus_{i=1}^{m} vv_{\mathcal{C}}^{(i)}$				
$(..., \{VV_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, ...) \leftarrow BS$ asserts $VV_{\mathcal{C}} = VV'_{\mathcal{C}}$				

Protocol 10: Voting phase (1/2) where the voter casts their vote. The vote and the vote authentication are stored on the bulletin board (BB), and the control components (CC) send back a verification for the received vote.

Voting phase (2/2) The voter confirms their vote (see Protocol 11).

When the first part of the voting phase has been successful, the voter extracts the confirm authentication CA from the ballot sheet, and sends it together with their Id to the bulletin board.

The bulletin board ensures the authentication is valid, otherwise the request is simply ignored. Then the bulletin board checks the voter has not already confirmed the vote, otherwise the bulletin board responds with the partial vote confirmations. If both checks succeed, the vote is extracted from the cast votes, and marked as confirmed. When the voter has not cast a vote yet, instead the empty vote is added to the confirmed votes.[18]

[18] If this happens, the voter has clearly made a mistake: They input their confirmation

Each control component extracts the partial confirm verification cv and then sends it to the bulletin board. The bulletin board forwards all to the voter.

The voter extracts the expected confirm verification CV out of the ballot sheet, and ensures the combination of the partial confirm verifications result in the same value. If yes, the voter has successfully confirmed the vote and is done. If no, the voter calls the election hotline to invalidate the vote.

Voter		**BB**		$\mathbf{CC}^{(i)}$ $i \in [1, m]$
knows BS		knows $\boldsymbol{hCA}, \boldsymbol{\mathcal{C}}_{Cast}$		knows $\boldsymbol{b}^{(i)}$
$(Id,...,CA,...) \leftarrow BS$				
	$\xrightarrow{Id, CA}$			
		asserts $(Id, \mathsf{Hash}(CA)) \in \boldsymbol{hCA}$ asserts $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf}$		
		$\mathcal{C} \leftarrow (Id, .) \in \boldsymbol{\mathcal{C}}_{Cast}$ $\boldsymbol{\mathcal{C}}_{Conf} \leftarrow \boldsymbol{\mathcal{C}}_{Conf} \cup \{(Id, \mathcal{C})\}$		
				$(..., cv^{(i)}, ...) = b_{Id}^{(i)}$
			$\xleftarrow[\text{A}]{cv^{(i)}}$	
		$\boldsymbol{cv} \leftarrow \{cv^{(1)},..., cv^{(m)}\}$		
	$\xleftarrow{\boldsymbol{cv}}$			
$\{cv^{(1)},..., cv^{(m)}\} \leftarrow \boldsymbol{cv}$ $CV' = \oplus_{i=1}^{m} cv^{(i)}$				
$(..., CV, ...) \leftarrow BS$ asserts $CV = CV'$				

Protocol 11: Voting phase (2/2) where the voter confirms their vote. The confirmation authentication is stored on the bulletin board (BB), and the control components (CC) send back a verification that the confirmation has been received.

7.2.3 Tally phase

The confirmed votes are tallied.

The control components perform the verifiable decryption-shuffle, and publish corresponding proofs and partial results on the bulletin board. After

authentication before verifying the vote verifications.

each control component verified the proofs of the other control components, the plain votes are counted and the result is announced.

Tally phase (1/2) In the first part of the tally phase, the control components verifiably shuffle and decrypt the votes (see Protocol 12).

The first control component maps the confirmed votes to their appropriate ciphertext counterpart. Then it shuffles the ciphertexts, and does the partial decryption. The proofs of both procedures and the intermediate result is posted on the bulletin board.

The next control component ensures the proofs are valid, else it aborts.[19] Then it does its own shuffle and partial decryption, again posting the result on the bulletin board. One after the other control component follows, always validating the proofs of all previous components. The last control component gets the plain votes as a result.

For the plain votes not to be attributable to individual voters, the shuffling requires at least two values to operate on (if only a single ciphertext is shuffled, the resulting plain vote obviously belongs to that ciphertext). The concrete shuffle implementation might enforce a higher lower bound. We rely on the authorities to choose an appropriate number of vote(r)s such that sufficient votes are handed in.[20]

For an election with n selectable voting options, the shuffle operates on n times the ciphertexts. This might negatively affect performance. One may therefore decide to perform a separate shuffle for each $\mathbb{C}$ subgroup. Each individual shuffle then handles less ciphertexts, while the anonymity set is as big as before.

The shuffling can be performed jointly over multiple instances, as long as the used $\mathcal{P}$'s are unambiguous and the same shuffle key pairs were chosen. This setup is advisable if voters of different eligibilities are active in the same voting circle: While for each eligibility, a separate instance needs to be setup, as in the end all votes are again shuffled together, the intended size of the anonymity set (given by the voting circles) is preserved.

[19] Note that each control component establishes the original list of ciphertexts by themselves, ensuring all control components have the same view of confirmed votes.

[20] In Switzerland, the administration defined voting cycles. For each voting cycle, only the aggregated results are published.

$\mathbf{CC}^{(i)}$
$i \in [1, m]$

knows $sk_S^{(i)}$

$\boldsymbol{\mathcal{E}}_0 \leftarrow \mathsf{MapCtEs}(\boldsymbol{\mathcal{C}}_{Conf}, \boldsymbol{CtE})$

$\forall j \in [1, i-1].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{j-1}, \boldsymbol{\mathcal{E}}_j, \pi_j)$
$(\boldsymbol{\mathcal{E}}_i, \pi_i) \xleftarrow{r} \mathsf{SShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{i-1}, sk_S^{(i)})$

publish $\boldsymbol{\mathcal{E}}_i, \pi_i$ on **BB**

Protocol 12: Tally phase (1/2) of control components where they first ensure the proofs of previous invocations are valid, then shuffle and partially decrypt the confirmed votes while storing proofs on the bulletin board (BB).

Tally phase (2/2) The control components verify the remaining proofs (see Protocol 13).

When the shuffle-decryption has been executed by all control components, each control component goes through all the proofs of the control components they have not verified yet.

When all control components have verified the proofs successfully, the plain votes are counted and the result is announced.

$\mathbf{CC}^{(i)}$
$i \in [1, m]$

$\forall j \in [i+1, m].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{j-1}, \boldsymbol{\mathcal{E}}_j, \pi_j)$

Protocol 13: Tally phase (2/2) where the control components (CC) verify the shuffle and decryption proofs on the bulletin board (BB) they have not checked yet in Protocol 12.

7.3 Algorithms

We now give pseudo-code for all syntax with is part of the core protocol. For the syntax belonging to the verifiable shuffle (prefixed with an S), see section 7.1.2.

The core idea is that the ballots are generated jointly by the control components, with the setup authority essentially only combining the randomness of all components. The combining operations are modular addition, XOR or chained permutations, with these operations all having the property that if at least one involved value is uniform at random, then the result will be.

We use modular addition to derive authentication secrets. These have to be brute-force safe. We denote the group where the authentication secrets are chosen from as $\mathbb{Z}_a$.

We further use modular addition to derive the verification secrets. These are shown to the user, and the attacker only has a single try to get it right, for the user not to suspect foul play. Per our assumption of a covert adversary, the probability to guess the right verification is therefore related to the adversary's deterrence factor.[21] We denote the group where the verification secrets are chosen from as $\mathbb{Z}_v$.

We chain permutations together to permute the PtC into the voter-specific PtC_{Id}. The permutations must be defined over all $\mathcal{C} \in \mathbb{C}$, and must be a bijection. The partial permutations always have to be applied in the same order to result in the same combined permutation. We denote the group where permutations are chosen from as $\mathbb{P}_{|\mathbb{C}|} : \mathbb{C} \to \mathbb{C}$.

Finally, we choose randomness for the verifiable shuffle to encrypt the votes. We denote where the randomness for a single encryption call is chosen from as $\{0,1\}^s$. We use XOR to combine the randomness from multiple control components, denoted as $\otimes$.

[21] This typically results in much shorter verification codes than the brute-force safe authentication codes.

Algorithm: GenIds(Id_1, n)
Input: First Id: $Id \in \mathbb{N}$
Number of Ids to generate: $n \in \mathbb{N}$

for $i \in [1, n-1]$ **do**
  $Id_i \leftarrow Id_1 + i$
$\boldsymbol{Id} \leftarrow (Id_1, ..., Id_n)$

return $\boldsymbol{Id}$

Algorithm 7.1: Generates n ids.

Algorithm: GenBallots($\boldsymbol{Id}$)
Input: Ids to generate ballots for: $\boldsymbol{Id} = (Id_1, ..., Id_{|\mathbb{V}|})$

for $Id \in \boldsymbol{Id}$ **do**
  for $\mathcal{C} \in \mathbb{C}$ **do**
    $vv_{\mathcal{C}} \xleftarrow{r} \mathbb{Z}_v$
    $r_{\mathcal{C}} \xleftarrow{r} \{0,1\}^s$
  $\boldsymbol{vv} \leftarrow \{vv_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}$
  $\boldsymbol{r} \leftarrow \{r_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}$

  $va \xleftarrow{r} \mathbb{Z}_a$
  $ca \xleftarrow{r} \mathbb{Z}_a$
  $cv \xleftarrow{r} \mathbb{Z}_v$

  $p \xleftarrow{r} \mathbb{P}_{|\mathbb{C}|}$

  $b_{Id} \leftarrow (va, \boldsymbol{vv}, ca, cv, p, \boldsymbol{r})$
$\boldsymbol{b} \leftarrow \{(Id, b_{Id}) \mid Id \in \boldsymbol{Id}\}$

return $\boldsymbol{b}$

Algorithm 7.2: Generates a ballot for each id.

Algorithm: MergeBallots($\boldsymbol{Id}, \boldsymbol{b}$)
Input: Ids to generate ballots for: $\boldsymbol{Id} = (Id_1,...,Id_{|\mathbb{V}|})$
Ballots $\forall i \in [1,m].CC_i$: $\boldsymbol{b} = \{(b_{Id}^{(i)}, Id) \mid Id \in \boldsymbol{Id}, \forall i \in [1,m]\}$

for $Id \in \boldsymbol{Id}$ **do**
  $\forall i \in [1,m].(va^{(i)}, \boldsymbol{vv}^{(i)}, ca^{(i)}, cv^{(i)}, p^{(i)}, \boldsymbol{r}^{(i)}) \leftarrow b_{Id}^{(i)}$
  for $\mathcal{C} \in \mathbb{C}$ **do**
    $VV_{\mathcal{C}} \leftarrow \oplus_{i=1}^{m} vv_{\mathcal{C}}^{(i)}$
    $R_{\mathcal{C}} \leftarrow \otimes_{i=1}^{m} r_{\mathcal{C}}^{(i)}$
  $\boldsymbol{VV} \leftarrow \{VV_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}$
  $\boldsymbol{R} \leftarrow \{R_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}$
  $VA \leftarrow \oplus_{i=1}^{m} va^{(i)}$
  $CA \leftarrow \oplus_{i=1}^{m} ca^{(i)}$
  $CV \leftarrow \oplus_{i=1}^{m} cv^{(i)}$
  $P \leftarrow \prod_{i=1}^{m} p^{(i)}$
  $B_{Id} \leftarrow (VA, \boldsymbol{VV}, CA, CV, P, \boldsymbol{R})$
$\boldsymbol{B} \leftarrow \{(Id, B_{Id}) \mid Id \in \boldsymbol{Id}\}$

return $\boldsymbol{B}$

Algorithm 7.3: Merges ballots generated by Algorithm 7.2.

Algorithm: HashAuths($\boldsymbol{Id}, \boldsymbol{B}$)
Input: Ids to hash auths for: $\boldsymbol{Id} = (Id_1,...,Id_{|\mathbb{V}|})$
Ballots: $\boldsymbol{B} = \{(B_{Id}, Id) \mid Id \in \boldsymbol{Id}\}$

for $Id \in \boldsymbol{Id}$ **do**
  $(VA,...,CA,...) \leftarrow B_{Id}$
  $hVA_{Id} \leftarrow \mathsf{Hash}(VA)$
  $hCA_{Id} \leftarrow \mathsf{Hash}(CA)$
$\boldsymbol{hVA} \leftarrow \{(Id, hVA_{Id}) \mid Id \in \boldsymbol{Id}\}$
$\boldsymbol{hCA} \leftarrow \{(Id, hCA_{Id}) \mid Id \in \boldsymbol{Id}\}$

return $(\boldsymbol{hVA}, \boldsymbol{hCA})$

Algorithm 7.4: Creates hashes of the authentication secrets.

Algorithm: GenCtEs($\boldsymbol{Id}, \boldsymbol{B}$, pk_S)
Input: Ids to generate CtE for: $\boldsymbol{Id} = (Id_1,...,Id_{|\mathbb{V}|})$
Ballots: $\boldsymbol{B} = \{(B_{Id}, Id) \mid Id \in \boldsymbol{Id}\}$
Shuffle public key: pk_S

for $Id \in \boldsymbol{Id}$ **do**
  $(..., P, \boldsymbol{R}) \leftarrow B_{Id}$
  for $\mathcal{C} \in \mathbb{C}$ **do**
    $\mathcal{C}' \leftarrow \mathcal{C} * P^{-1}$
    $\mathcal{P} \xleftarrow{.} (., \mathcal{C}') \in PtC$
    $\mathcal{E}_{\mathcal{C}} \leftarrow \mathsf{SEnc}(R_{\mathcal{C}}, pk_S, \mathcal{P})$
  $CtE_{Id} \leftarrow \{(\mathcal{C}, \mathcal{E}_{\mathcal{C}}) \mid \mathcal{C} \in \mathbb{C}\}$
$\boldsymbol{CtE} \leftarrow \{(Id, CtE_{Id}) \mid Id \in \boldsymbol{Id}\}$
return $\boldsymbol{CtE}$

Algorithm 7.5: Generates the lookups to derive the shuffle input.

Algorithm: GenBallotSheets($\boldsymbol{Id}, \boldsymbol{B}, \mathbb{V}, PtC$)
Input: Ids to hash auths for: $\boldsymbol{Id} = (Id_1,...,Id_{|\mathbb{V}|})$
Ballots: $\boldsymbol{B} = \{(B_{Id}, Id) \mid Id \in \boldsymbol{Id}\}$
Electorate: $\mathbb{V} = (V_1,..., V_{|\mathbb{V}|})$
Plain to Code lookup: PtC

$i = 1$
for $Id \in \boldsymbol{Id}$ **do**
  $(VA, \boldsymbol{VV}, CA, CV, P, \boldsymbol{R}) \leftarrow B_{Id}$
  $PtC_{Id} \leftarrow PtC * P$
  $BS_i \leftarrow (Id, VA, \boldsymbol{VV}, CA, CV, PtC_{Id}, \boldsymbol{R})$
  $i \leftarrow i + 1$
$\boldsymbol{BS} \leftarrow \{(V_i, BS_i) \mid i \in \{1, \ldots, |\mathbb{V}|\}$
return $\boldsymbol{BS}$

Algorithm 7.6: Generates the ballot sheets.

Algorithm: MapCtEs($\boldsymbol{\mathcal{C}}_{Conf}$, $\boldsymbol{CtE}$)
Input: Confirmed Votes: $\boldsymbol{\mathcal{C}}_{Conf}$
Code to Encryption lookup: $\boldsymbol{CtE}$

for $(Id, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf}$ **do**
  $CtE_{Id} \xleftarrow{\cdot} (Id, .) \in \boldsymbol{CtE}$
  $\mathcal{E}_{\mathcal{C}} \xleftarrow{\cdot} (\mathcal{C}, .) \in CtE_{Id}$
  $\boldsymbol{\mathcal{E}} \leftarrow \boldsymbol{\mathcal{E}} \cup \{\mathcal{E}_{\mathcal{C}}\}$

return $\boldsymbol{\mathcal{E}}$

Algorithm 7.7: Maps the confirmed code votes to encryptions.

7.4 Extensions

The proposed protocol reaches all security properties under the stated trust assumptions, as we proof in chapter 9. However, there are extensions we might incorporate into the protocol to further harden the system, at the price of increased complexity and/or reduced transparency.

Shuffle at the printer Instead of the setup authority preparing the ballot sheets with a corresponding voter, the ballot sheets could at that step still remain separated from individual voters. Only at the printer authority do we assign the ballot sheets to voters, for example using a physical process which makes it impossible to store the correspondence. This increases privacy against an attacker which has access to the setup authority, but is not able to attribute requests to the bulletin board to individual voters.

Concretely, we establish the strong notion of everlasting privacy, as even before the verifiable shuffle[22], the votes are not attributed to their voter.

Keep bulletin board partially or fully secret Instead of exposing the full bulletin board to the public, we could instead opt to expose only selected entries. For an attacker which has only the partial public view of the bulletin board, we can strengthen our security properties.

If the bulletin board hides any values related to the verifiable shuffle, our protocol achieves the very strong notion of everlasting privacy. Because the voting codes are effectively perfect encryption, no cryptographic assumption are used any more to protect vote secrecy.

If the bulletin board hides the voting codes / vote verification exchange, the protocol achieves receipt freeness for voters which are compromised only after they have confirmed their vote. These voters can no longer proof to the attacker which vote is represented on the bulletin board.

Use a pseudo-random generator for the shuffle randomness Instead of the control components picking an r for each voting code (merged into R, then used for the SEnc call in $\mathsf{GenCtEs}$), we could instead pick a single random seed to initialize an appropriate pseudo-random generator.[23]

This reduces the randomness picked at the control components by a substantial amount, especially when many voting options are possible.

[22] Which currently breaks this notation as it likely uses cryptographic assumptions.

[23] Both mechanisms are per-voter as the randomness is included on the ballot sheet.

7.5 Summary

Based on the setting and the iterative proposal (see chapter 5 and chapter 6), we formally defined a protocol for vote électronique. We clearly established the trust assumptions it requires and the security properties it achieves, which we formally proof in chapter 9.

First, we clearly described the infrastructure we depend on. We require authentic channels throughout the execution of the protocol, and the stronger secure channels only in the setup phase. Based on the authentic channels, we specify our append-only storage we name the bulletin board. We define clear syntax and a correctness definition for the verifiable decryption-shuffle we need in the tally phase of the protocol. We also assume an election hotline, which resolves conflicts impossible within the cryptographic protocol.

Based on the infrastructure, we introduce the protocol divided in setup, voting and tally phases. The setup phase establishes key material with the control components, and ensures the bulletin board and each voter receives appropriate material for the voting phase to start. In the voting phase, the voter casts and confirms their vote, verifying each round-trip with the control components. In the tally phase, the control components verifiably shuffle and decrypt the votes.

We also give pseudo-code for all syntax which is part of our core protocol (which is everything except the infrastructure). An informal security analysis follows, as well as a short discussion about possible extensions of the protocol to further decrease the trust assumptions or strengthen the security properties.

Part III

Proof

Chapter 8

Formal Model

We first define an abstract voting model and its syntax. We use that syntax to derive game-based definitions for our security properties. Defining the security properties in the abstract makes discussing them easier.

This model can then be instantiated by some concrete voting protocol (for example, our proposal of chapter 7).

8.1 Bulletin-Board Voting

We call our model *Bulletin-Board Voting*, as the component at the centre of our model is a bulletin board: A public append-only storage over which the other involved roles interact.

Roles We introduce the few roles we need to describe our properties. We refer to instances of roles as honest if they follow the protocol exactly and do not leak secrets, and as dishonest if they are under control of the adversary.

The *Voters* (**V**) are eligible to cast and confirm votes in the election. For all honest voters, we require our properties to hold, even if dishonest voters also participate in the protocol.

The *Control Components* (**CC**) are servers run on behalf of the election authorities. We assume a single control component to be honest, but allow further, dishonest, control components. All of our game-based definitions include the control components.

The *Bulletin Board* (**BB**) is a public append-only storage. We assume all communication between the voters and control components goes over the bulletin board. The bulletin board is assumed honest.[1]

[1] Depending on the concrete voting protocol, the bulletin board's honestly could be enforced by the control components.

The *Adversary* ($\mathcal{A}$) exists only once and remembers state between invocations. The adversary does not learn secrets and cannot change execution of the honest roles, but is otherwise able to to anything (like creating or changing messages sent between roles, or altering execution of dishonest roles).[2]

Protocol principles We avoid defining details about the voting protocol itself at all, as we do not require any such assumptions for our security definitions. We have to however clear up some terminology to precisely describe the properties.

We divide the protocol into setup, voting and tally phases. During the setup phase, key material is prepared necessary for voting and tally (but no votes are cast yet). In the voting phase, the voters cast and confirm their votes. In the tally phase, the confirmed votes are counted and the result is published.

In the voting phase, the voter first casts the vote. The vote represents some selection of voting option(s). Depending on the protocol, the voter may or may not exchange some messages with the control components (via the bulletin board). If the voter is happy with how the vote was cast, they confirm their vote. Again, some additional messages with the control components may be exchanged. If the voter is happy with how the vote was confirmed, their *Individual Verifiability Check* succeeds. This specifically means that the voter is convinced that their cast and confirmed vote was received by the control components exactly as intended.[3]

Syntax by protocol phase To precisely describe our properties, we require to define some syntax.

We define a function Setup which encapsulates the setup phase. Given *params* (which describes necessary election configuration like participating voters, voting options, cryptographic parameter sizes, ...), it returns a list of voter states v, a list of control component states cc and the initial bulletin board state bb. It might be run under adversarial influence which we denote as $\mathsf{Setup}_{\mathcal{A}}$.

For the voting phase, we assume no syntax to keep the definitions simple, and to not artificially restrict the protocols our definitions apply to. However, concrete protocols have to provide oracles to the adversary for each

[2] Likely, the concrete protocol will further restrict the behaviour of the adversary with some additional assumptions, for example that the adversary is polynomially bound or cannot interfere with messages sent over secure channels.

[3] Notice the gap between a voter who confirmed their vote, but was not happy with the messages exchanged afterwards such that the individual verifiability check failed: Is the vote now confirmed by the control components as intended? In general, this cannot be resolved (like the two generals' problem), and we therefore leave it unspecified in our model.

type of request answered by honest roles. The adversary then drives the voting phase given all these oracles, possibly calling them in any order and given any arguments. The proofs have to reflect this still does not break the to-be-shown security property. We write $\mathcal{A}^{\text{Role}}$ to give the adversary access to all oracles provided by the honest *Role*.

For the tally phase, we define a function $\mathsf{Tally}()$. It returns the election result, interacting with the other instances. It might be run under adversarial influence which we denote as $\mathsf{Tally}_{\mathcal{A}}$.

Syntax to query protocol state To define our properties, we require some syntax to query the state of instances.

We define the following functions to assess the state of the voter:

- $\mathsf{Confirmed}(\mathrm{V})$ returns true if the voter V has confirmed their cast vote, otherwise false.
- $\mathsf{IVCheck}(\mathrm{V})$ returns true if the voter V has accepted their individual verifiability check, otherwise false.[4]

We further define the following functions to assess the state represented on the bulletin board:

- $\mathsf{Count}(\mathrm{BB})$ returns the number of confirmed votes on the bulletin board BB.
- $\mathsf{Extract}(\mathrm{BB}, i)$ returns the selection represented by the confirmed vote of voter V_i on the bulletin board BB.[5] If no confirmed vote is on the bulletin board, it returns $\perp$. If multiple confirmed votes are on BB, a list is returned.
- $\mathsf{Validate}(\mathrm{BB}, i)$ returns true if there is a confirmed vote of voter V_i on the bulletin board BB and the selection represented by said confirmed vote is valid, otherwise false.
- $\mathsf{CorrectTally}(\mathrm{BB})$ tallies the selections represented by the confirmed votes on the bulletin board and returns the result.[6]
- $\mathsf{VerifyTally}(\mathrm{BB}, t)$ returns true if the bulletin board BB could represent some claimed election result t, otherwise false.

Given the same arguments, we assume the functions to always return the same result. While the concrete protocol has to be able to implement these functions correctly, it does not necessarily need to do so in an efficient manner.

[4] Note that $\mathsf{IVCheck}(\mathrm{V}) \implies \mathsf{Confirmed}(\mathrm{V})$.

[5] Notably independent whether the selection is valid.

[6] Note that this might different than the result returned by the $\mathsf{Tally}_{\mathcal{A}}()$, in case the adversary is able to influence the result output in the tally phase.

8.2 Security properties

Given our syntax, we can now precisely define the security properties. We motivate them out of the definitions of our setting (see chapter 5), which themselves are extracted from law. We relate the security properties to their equivalents known by literature (if any), and provide precise game-based definitions. There are two previous proofs we know of in the Swiss context over internet voting schemes, and we describe difference of definitions in appendix D.

All properties together result in a secure voting system. Individual verifiability ensures votes are cast as intended. Authentication ensures only votes from eligible voters are confirmed. Universal verifiability then ensures the confirmed votes are counted appropriately. Throughout the process, vote secrecy and fairness hold for secret and fair elections.

Simplifications We aim to keep the definitions as simple as possible, to focus the attention on the to-be-achieved security properties rather than on notational technicalities. We choose to restrict the adversary in two cases. The concrete protocol to which the definition is applied has to of course argue that these restrictions do not result in an insecure real-world system, or remove them from the definition.

We restrict the adversary when it is given the power to choose the honest voter's selection: The adversary has to do so at the start of the voting phase, hence before first learning (additional) private state of compromised instances or interacting with the protocol. This is a restriction of the adversary we argue natural: An honest voter decides its selection independently to the concrete run of the protocol. We incorporate this restriction as it makes our definitions much simpler.

We further restrict the adversary by not giving it the power to decide which voter and which control component is honest: We fix the honest control component to CC_1, and the honest voter to V_1, while all other instances are assumed under control of the adversary. This avoids us having to model the adversary choosing the honest roles, possibly even only after interacting with the protocol for some time. This is a restriction we only incorporated for simplicity, and we propose that protocols using the definitions either give the choice back to the adversary, or argue why fixing the honest instances does not incur a loss in generality.

Notation Honest instances of roles are created with $\xleftarrow{c}$ out of an appropriate initial state, like $\mathrm{V} \xleftarrow{c} \mathbf{V}(v)$.

We write $\boldsymbol{L}_{\ominus 1}$ if we want to exclude the first element of the list $\boldsymbol{L}$.

To formulate the winning conditions in the games, we use *assert* statements (rather than returning a value depending on some condition). This simplifies understanding the property, as it is immediately clear what the property asserts.

8.2.1 Individual Verifiability

We defined individual verifiability like this:

> Individual Verifiability holds when voters are given exactly one of two proofs: Voters who participate electronically are given a proof that the vote has been registered successfully by the server, exactly as cast. Voters who did not participate electronically can request a proof that their vote has not been registered by the server.

This definition is composed out of two requirements: One for voters which indeed do participate, and another one for voters which do not participate. We introduce these properties as individual verifiability and participation verifiability.

Both these requirements are defined relative to "successfully registered" votes, which itself is composed out of two other properties. Voters must only cast and confirm a single vote, and only votes which represent a valid selection must be stored by the server. We define the former as eligibility uniqueness in section 8.2.4, and the latter as vote verifiability here.

Individual Verifiability In our precise game definitions here, we refer only to the first part as *Individual Verifiability*, consistent with the use of this term in the literature. For this property to hold, the individual verifiability check for the voter may only pass if the confirmed vote has been included on the bulletin board.

We first setup the protocol using $\mathsf{Setup}_{\mathcal{A}}$. The adversary then chooses the selection of the honest voter. Afterwards, the adversary plays against this honest voter, the honest control component and the bulletin board, while being in control of all other instances. In the end, the adversary wins if it can break the valid state as defined by the assert statement.

The adversary wins the game if the individual verifiability check succeeds although the bulletin board does not include a confirmed vote, or includes a confirmed vote representing a different selection. We define the advantage of an adversary against individual verifiability as:

$$\mathbf{Adv}^{\mathrm{IV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{IV}}_{\mathcal{A}} \text{ assert fails}]$$

$\mathbf{Exp}^{\mathrm{IV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

$s \leftarrow \mathcal{A}()$
$\mathrm{V}_1 \xleftarrow{c} \mathbf{V}(v_1, s), \mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{V}_1, \mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

$\mathsf{assert\ IVCheck}(\mathrm{V}_1) \implies \mathsf{Extract}(\mathrm{BB}, 1) = s$

Game 1: Game for individual verifiability. We let the adversary choose s, the selection of the voter.

Participation Verifiability The second part of the definition we refer to as *Participation Verifiability*, a new term, as we are not aware of this property being used in the literature. It allows voters to verify whether they have participated or not after the voting phase has finished. We explicitly formulate it more general, requiring the check to succeed for all eligible voters. Defining it only for abstaining eligible voters does not make much sense when the check is implemented it practice (chicken-egg problem).

Compared to the previous game, the adversary may now additionally decide whether the voter should abstain. The adversary can therefore either attack the first or the second assert statement.

$\mathbf{Exp}^{\mathrm{PV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

$s \leftarrow \mathcal{A}()$
$\mathrm{V}_1 \xleftarrow{c} \mathbf{V}(v_1, s), \mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{V}_1, \mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

$\mathsf{Tally}_{\mathcal{A}}()$

$\mathsf{assert\ IVCheck}(\mathrm{V}_1) \implies \mathsf{Extract}(\mathrm{BB}, 1) = \top$
$\mathsf{assert\ not\ Confirmed}(\mathrm{V}_1) \implies \mathsf{Extract}(\mathrm{BB}, 1) = \bot$

Game 2: Game for participation verifiability. We let the adversary choose s, the selection of the voter or $\bot$ for an abstaining voter.

The adversary wins the game if the individual verifiability check succeeded but no vote is confirmed on the bulletin board. The adversary further wins the game if the voter did not confirm their vote but some confirmed vote corresponding to that voter is included on the bulletin board. We define the advantage of an adversary against participation verifiability as:

$$\mathbf{Adv}^{\mathrm{PV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{PV}}_{\mathcal{A}} \text{ assert fails}]$$

Instead of having two assert statements in the same game, we could also instead define two games, each only featuring a single assert statement. We name the game variant with only the first assert statement *Participation Verifiability - Participating Voter*, with the game $\mathbf{Exp}_{\mathcal{A}}^{\text{PVP}}$. We name the game variant with only the second assert statement *Participation Verifiability - Abstaining Voter*, with the game $\mathbf{Exp}_{\mathcal{A}}^{\text{PVA}}$.

The adversary against participation verifiability wins if it wins in either game variant, hence we can upper bound the advantage by the sum of the advantages of the two game variants.

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PV}} \leq \mathbf{Adv}_{\mathcal{A}}^{\text{PVP}} + \mathbf{Adv}_{\mathcal{A}}^{\text{PVA}}$$

As defined, participation verifiability - participating voter is a weaker property than individual verifiability. We can therefore bound the advantage of an adversary against $\mathbf{Exp}_{\mathcal{A}}^{\text{PVP}}$ by the advantage of an attacker against $\mathbf{Exp}_{\mathcal{A}}^{\text{IV}}$.

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PVP}} \leq \mathbf{Adv}_{\mathcal{A}}^{\text{IV}}$$

Further, as defined, participation verifiability - abstaining voter is the same property as eligibility verification. We can therefore equal the advantage of an adversary against $\mathbf{Exp}_{\mathcal{A}}^{\text{PVA}}$ to the advantage of an attacker against $\mathbf{Exp}_{\mathcal{A}}^{\text{EV}}$.

$$\mathbf{Adv}_{\mathcal{A}}^{\text{PVA}} = \mathbf{Adv}_{\mathcal{A}}^{\text{EV}}$$

Vote Verifiability *Vote Verifiability* ensures votes represent a valid selection. Depending on the precise voting scheme, there might be restrictions which voting choices can be taken at the same time, or whether single or multiple voting options have to be selected. Getting this right ensures each vote counts only once (no double-selection within the same vote) and votes do not corrupt other votes (for example, by selecting negatively[7]).

In this game, the adversary gets the secrets of all voters, and must attempt to cast and confirm an invalid vote.

The adversary wins the game if some confirmed vote is on the bulletin board, but the selection it represents is invalid. We define the advantage of an adversary against vote verifiability as:

$$\mathbf{Adv}_{\mathcal{A}}^{\text{VV}} = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{VV}} \text{ assert fails}]$$

[7] Like selecting a specific candidate -1 times.

$\mathbf{Exp}^{\mathrm{VV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$
$\mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}, \boldsymbol{cc}_{\ominus 1})$
assert $\mathsf{Extract}(\mathrm{BB}, 1) = \top \implies \mathsf{Validate}(\mathrm{BB}, 1)$

Game 3: Game for vote verifiability.

8.2.2 Universal Verifiability

We defined universal verifiability like this:

> Universal verifiability holds when the auditors are given a proof that the result is composed out of all, and only of, successfully registered votes.

Universal verifiability is defined relative to "successfully registered" votes. We already have a precise understanding of what this refers to (see section 8.2.1), and all confirmed votes on the bulletin board fulfil this definition (guaranteed by the individual verifiability and authentication properties). The bulletin board further guarantees that votes, once included, are unchanged. Therefore, after the voting phase, the bulletin board contains *"all, and only (...), successfully registered votes"*.

Consequentially, our property needs only to prove that the bulletin board as given after the voting phase is tallied correctly.

Universal Verifiability *Universal Verifiability* ensures all confirmed votes included on the bulletin board are tallied correctly.

We let the adversary interact arbitrarily with the bulletin board and the honest control component. Then, the adversary needs to produce some claimed election result which cannot be refuted, but is still wrong.

$\mathbf{Exp}^{\mathrm{UV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$
$\mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}, \boldsymbol{cc}_{\ominus 1})$
$t \leftarrow \mathsf{Tally}_{\mathcal{A}}()$
assert $\mathsf{VerifyTally}(\mathrm{BB}, t) \implies \mathsf{CorrectTally}(\mathrm{BB}) = t$

Game 4: Game for universal verifiability. We let the adversary choose t', the claimed voting result.

The adversary wins the game if the adversary-chosen t validates relative to the bulletin board, although the actual result represented on the bulletin board is different to t. We define the advantage of an adversary against universal verifiability as:

$$\mathbf{Adv}^{\mathrm{UV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{UV}}_{\mathcal{A}} \text{ assert fails}]$$

8.2.3 Vote Secrecy and Fairness

We defined vote secrecy and fairness like this:

> Vote secrecy holds if the plain vote cannot be attributed to the voter. Fairness ensures the adversary does not learn partial election results before the official tally.

For the privacy properties, we use indistinguishability games: The adversary has to be unable to distinguish two different worlds from each other. In the first world with challenge bit 0, the adversary sees the real world where the honest voter casts their real vote. In the second world with challenge bit 1, the adversary sees a simulated world where the honest voter casts some fake vote. After the game, the adversary has to guess the challenge bit.

Vote secrecy requires votes either to be encrypted or not attributable to voters (e.g. by using anonymous channels). When a protocol goes for the latter, it might even allow the voter to cast plain votes. However, we argue protocols which allow to cast plain votes to be unrealistic: It would need strong trust assumptions on an anonymous and confidential channel from the voter to the bulletin board (for vote secrecy and fairness), and strong trust assumption on said bulletin board in order to keep the votes private until the tally phase (for fairness). Here, we therefore define vote secrecy with some restriction in generality; specifically we assume the cast votes to be encrypted. This simplifies the vote secrecy definition, and even allows us to argue that vote secrecy implies fairness.

As both of the vote secrecy as well as the fairness definition are somewhat novel, we argue more explicitly why they are appropriate in appendix E.

Vote secrecy *Vote Secrecy* ensures the adversary does not learn the voter selection.

We formalize this notion in an indistinguishability game. We let the adversary decide upon both the real as well as the fake selection of the voter, and the voter casts then depending on the challenge bit. The voting phase proceeds as usual, with the attacker being able to freely interact with the honest instances, and having full control over the dishonest instances.

In the end, always the real votes are counted; with challenge bit 0 the tally is executed normally, with challenge bit 1 the tally is simulated as if the voter would have cast their real selection. This prevents the adversary from trivially distinguishing the real and the fake world by looking at the tally result.

Finally, the adversary is given the tally result r and then has to output its challenge bit guess.[8]

We extend our syntax with the function $\mathsf{SimTally}_{\mathcal{A}}()$ which modifies the tally in such a way that the vote of the honest voter is simulated to represent $s_{1-\beta}$.

$\mathbf{Exp}^{\mathrm{VS},\beta}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

$(s_0, s_1) \leftarrow \mathcal{A}()$
$\mathrm{V}_1 \xleftarrow{c} \mathbf{V}(v_1, s_\beta), \mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{V}_1,\mathrm{CC}_1,\mathrm{BB}}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

if $\beta = 0$ then $r \leftarrow \mathsf{Tally}_{\mathcal{A}}()$ else $r \leftarrow \mathsf{SimTally}_{\mathcal{A}}()$
$d \leftarrow \mathcal{A}^{\mathrm{V}_1,\mathrm{CC}_1,\mathrm{BB}}(r)$

if $d = \beta$ then return 1 else return 0

Game 5: Game for vote secrecy. We let the adversary choose s_0 and s_1, the possible selections of the voter, of which the voter casts s_β.

The adversary wins the game if it can distinguish whether the fake or the real selection is cast by the voter. Further, the adversary wins if it can distinguish the simulated tally from the real tally. The adversary also wins if any of the oracles provided by the honest instances enable it to distinguish the real or the fake world.

The adversary wins the game if it can distinguish $\mathbf{Exp}^{\mathrm{VS},0}_{\mathcal{A}}(params)$ from $\mathbf{Exp}^{\mathrm{VS},1}_{\mathcal{A}}(params)$. We define the advantage of an adversary against vote secrecy as:

$$\mathbf{Adv}^{\mathrm{VS}}_{\mathcal{A}} = |\Pr[\mathbf{Exp}^{\mathrm{VS},0}_{\mathcal{A}}(params) = 1] - \Pr[\mathbf{Exp}^{\mathrm{VS},1}_{\mathcal{A}}(params) = 1]|$$

Fairness *Fairness* ensures the adversary does not learn partial election results before the official tally.

[8] When using verified decryption-shuffles, the result r would also include some proof Π of correct shuffle and decryption. Note that with challenge bit 1, Π needs to be simulated over the bulletin board the adversary sees, else distinguishing the real and the simulated tally might be trivial.

We formalize this notation similar to how we formalized the vote secrecy definition (see section 8.2.3). We again let the adversary decide upon both the real as well as the fake selection of the voter, and the voter casts then depending on the challenge bit. The voting phase proceeds as before, with the attacker being able to freely interact with the honest instances, and having full control over the dishonest instances.

However, then the adversary already has to output the guess for the challenge bit: Fairness only needs to hold until the tally phase.

$\mathbf{Exp}^{\mathrm{F},\beta}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$
$(s_0, s_1) \leftarrow \mathcal{A}()$
$\mathrm{V}_1 \xleftarrow{c} \mathbf{V}(v_1, s_\beta), \mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$d \leftarrow \mathcal{A}^{\mathrm{V}_1, \mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$
if $d = \beta$ then return 1 else return 0

Game 6: Game for fairness. We let the adversary choose s_0 and s_1, the possible selections of the voter, of which the voter casts s_β.

The adversary wins the game if it has a higher probability of winning the game than random guessing. We define the advantage of an adversary against fairness as:

$$\mathbf{Adv}^{\mathrm{F}}_{\mathcal{A}} = |\Pr[\mathbf{Exp}^{\mathrm{F},0}_{\mathcal{A}}(params) = 1] - \Pr[\mathbf{Exp}^{\mathrm{F},1}_{\mathcal{A}}(params) = 1]|$$

As defined, fairness is a weaker property than vote secrecy. We can therefore bound the advantage of an adversary against fairness by the advantage of an attacker against vote secrecy.

$$\mathbf{Adv}^{\mathrm{F}}_{\mathcal{A}} \leq \mathbf{Adv}^{\mathrm{VS}}_{\mathcal{A}}$$

8.2.4 Authentication

We defined authentication like this:

> Authentication holds when the adversary cannot insert votes without having control of the voter.

In the literature, the property described here is usually referred to as eligibility verification. The adversary should not be able to impersonate voters without having control of them, e.g. learning their secrets. In this section, we further include the requirement that voters must only cast and confirm

a single vote, which we refer to as eligibility uniqueness, as argued for in section 8.2.1.

Eligibility Verifiability *Eligibility Verifiability* ensures every vote considered in the tally is confirmed by an eligible voter.

We let the adversary choose the selection of the voter. Without the voter doing the confirmation, the adversary has to place a confirmed vote on the bulletin board.

$\mathbf{Exp}^{\mathrm{EV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$
$s \leftarrow \mathcal{A}()$
$\mathrm{V}_1 \xleftarrow{c} \mathbf{V}(v_1, s), \mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{V}_1, \mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$
assert not $\mathsf{Confirmed}(\mathrm{V}_1) \implies \mathsf{Extract}(\mathrm{BB}, 1) = \perp$

Game 7: Game for eligibility verifiability. We let the adversary choose s, the selection of the voter or $\perp$ for an abstaining voter.

The adversary wins the game if the voter has not confirmed their vote, although some corresponding vote is confirmed on the bulletin board. We define the advantage of an adversary against eligibility verifiability as:

$$\mathbf{Adv}^{\mathrm{EV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{EV}}_{\mathcal{A}} \text{ assert fails}]$$

Eligibility Uniqueness *Eligibility Uniqueness* ensures every eligible voter can only confirm a single vote.

We give the adversary access to all voter secrets. The adversary then has to insert multiple votes corresponding to the same voter.

$\mathbf{Exp}^{\mathrm{EU}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$
$\mathrm{CC}_1 \xleftarrow{c} \mathbf{CC}(cc_1), \mathrm{BB} \xleftarrow{c} \mathbf{BB}(bb)$
$\mathcal{A}^{\mathrm{CC}_1, \mathrm{BB}}(\boldsymbol{v}, \boldsymbol{cc}_{\ominus 1})$
assert $\forall i \in [1, |\boldsymbol{v}|].|\mathsf{Extract}(\mathrm{BB}, i)| \leq 1$

Game 8: Game for eligibility uniqueness.

The adversary wins the game if multiple confirmed votes corresponding to the same voter are confirmed on the bulletin board. We define the advantage of an adversary against eligibility verifiability as:

$$\mathbf{Adv}^{\mathrm{EU}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{EU}}_{\mathcal{A}} \text{ assert fails}]$$

8.3 Summary

We first defined a formal model we call the Bulletin Board model. It assumes voters, control components and the bulletin board to exist. We clear up some terminology, and then define syntax we require for our security properties. The model places very few assumptions, and should therefore be generally applicable.

Given the formal model, we then precisely defined the security properties. Starting from formulation as given by law, we decompose into clearly separated and easily definable security properties. Already in this abstract formal modal, we are able to show some relations between the properties which will simplify our concrete proofs later on.

Chapter 9

Computational Proof

We now proof the proposal of chapter 7 fulfils the claimed properties.

First, we make our infrastructure assumptions explicit, and describe how these assumptions are represented in our proofs. Afterwards, we apply the game-based definitions of chapter 8 to our proposal and the infrastructure. Clear bounds are established for the attacker to win each of these games.

In the verifiability games, the adversary targets to break some assertion. We continuously modify the games by inserting early aborts for cases where we can clearly bound the attacker advantage. After sufficient modifications, called game hops, we end up with a game that is impossible to win for the adversary. We then collect the bounds, hence sum up all bounded attacker advantages of our game hops, to end up with the overall attacker advantage.

In the privacy games, the adversary needs to differentiate two different worlds from each other. We start with the game in world 0, then continuously modify it for it to become more and more similar to the game in world 1. For each modification, again called game hop, we bound the probability that the adversary can distinguish the game before the modification from the game after the modification. When our modified game is exactly equal the game of the world 1, we again collect the bounds, hence sum up all bounded attacker advantages of our game hops to end up with the overall attacker advantage.

9.1 Infrastructure assumptions

In the proposal specification we clearly separated the core protocol from the infrastructure it relies upon. We will continue to do so in the proofs. In this section, we clearly describe what is assumed of the infrastructure and how it is therefore represented in the proofs. We collectively describe any assumptions part of this section as the *Infrastructure Assumptions*.

The infrastructure and the core protocol rely on cryptographic parameters such as group sizes. We assume these parameters are picked appropriately to reach the aimed for security level, and all participants of the protocol (and the infrastructure) have access to these values.

Channels The secure channels are used in the setup phase to exchange secret data between the control components to the setup authority, and from the setup authority via the printer to the voter. The authentic channels are used throughout the protocol to exchange data with the bulletin board.

We assume both the secret as well as the authentic channels to be perfect, hence the attacker has zero advantage when trying to break their properties.

In our computational proof, we do not mention the channels: Data exchanged over these channels simply continues to be processed within the game of the challenger.[1] When exchanged over authentic channels, the attacker will be given access to the values.

Hash The hash function Hash is used to authenticate voters.

We assume the hash function to be pre-image resistant, hence given h it is hard to find m such that $h = \mathsf{Hash}(m)$. The hash function supports input of arbitrary size.

Our computational proofs include the hash functions. We denote the advantage of an attacker $\mathcal{B}$ against pre-image resistance of Hash with $\mathbf{Adv}^{\mathrm{pIR}}_{\mathcal{B},\mathsf{H}}$.

Bulletin Board The bulletin board stores the progress of the protocol. Its processing is verified by the control components, which will detect any deviation from the specification, such as incorrectly executed computations, or changed or removed entries.

The bulletin board is untrusted. However, as it communicates state updates over the authentic channels (which we already assumed to exist), it still cannot misbehave: The honest control component will only accept messages sent over the authentic channel, which prevents the bulletin board from publishing inconsistent state.[2]

In our computational proofs, the bulletin board is fully under the control of the adversary (and therefore omitted as another explicit instance). This concretely means any communication in between the voter and the honest control component goes directly over the adversary. In turn, we make it

[1] We assumed in section 7.1 that values sent over authentic channels can also convince third parties of the respective sender, hence values can also be attributed when they are forwarded.

[2] As this would be detectable, and therefore outside the attacker model.

explicit how the honest control component re-executes all computations of the bulletin board (to ensure it does not misbehave). Our challenger game therefore does includes all computations specified for the bulletin board, as they are also executed by the honest control component.

Verifiable shuffle The verifiable shuffle consisting out of the five algorithms $(\mathsf{SKGen}, \mathsf{SKAgg}, \mathsf{SEnc}, \mathsf{SShuffleDecrypt}, \mathsf{SVerifyShuffleDecrypt})$ ensures the vote and the voter are separated before the vote is decrypted.

In the setup phase, we assume the following (reusing terminology of definition 7.1):

- $\mathsf{SKGen}()$ is one-way when executed honestly: Given pk_S, the adversary does not learn sk_S.
- $\mathsf{SKAgg}(\boldsymbol{pk}_S)$ is one-way: Given pk_S, the adversary does not learn the associated sk_S. Further, additionally given some decryption secret keys of pk_S, the attacker does not learn additional decryption secret keys. These properties hold as soon as some of the involved keys have been produced by an honestly executed SKGen.[3]
- $\mathsf{SEnc}(r, pk_S, m)$ is IND-CPA secure when executed honestly using uniform at random r: Given c and pk_S, the adversary does not learn what m it represents.

We denote the advantage of an attacker $\mathcal{C}$ against the one-wayness of SKGen and SKAgg with $\mathbf{Adv}^{\mathrm{KOW}}_{\mathcal{C},\mathsf{S}}$. We denote the advantage of an attacker $\mathcal{D}$ against IND-CPA security of SEnc with $\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{D},\mathsf{S}}$.

In the tally phase, we assume the following:

- $\mathsf{SShuffleDecrypt}(\mathbf{c}, sk_S)$ is IND-CPA secure: Given $\mathbf{c}$, sk_s, $\mathbf{c}'$ and π, the adversary does not learn what plaintext is represented (unless it was the last iteration, in which case the plaintext values are revealed trivially).
- $\mathsf{SShuffleDecrypt}(\mathbf{c}, sk_S)$ shuffles when executed honestly: Given $\mathbf{c}$, $\mathbf{c}'$ and π, the adversary cannot learn which $c \in \mathbf{c}$ corresponds to which $c' \in \mathbf{c}'$.
- $\mathsf{SVerifyShuffleDecrypt}(\mathbf{c}, \mathbf{c}', \pi)$ is correct when executed honestly: Given $\mathbf{c}$ the adversary cannot come up with $\mathbf{c}'$, π such that $\mathbf{c}'$ represents different plaintext as $\mathbf{c}$, but $\mathsf{SVerifyShuffleDecrypt}$ returns true.

We denote the advantage of an attacker $\mathcal{E}$ against IND-CPA security of $\mathsf{SShuffleDecrypt}$ with $\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{E},\mathsf{S}}$. We denote the advantage of an attacker

[3] Note that potentially malicious control components execute SKGen, hence they might execute it e.g. relative to some other secret keys or using bad randomness.

$\mathcal{F}$ against the shuffling of SShuffleDecrypt with $\mathbf{Adv}^{\text{shuffle}}_{\mathcal{F},\mathsf{S}}$. We denote the advantage of an attacker $\mathcal{G}$ against the correctness of SVerifyShuffleDecrypt with $\mathbf{Adv}^{\text{verify}}_{\mathcal{G},\mathsf{S}}$.

Election hotline Voters can contact the election hotline when they have confirmed their vote, but have not received the correct confirm verification.

We assume however the election hotline resolves these conflicts results in no advantage to the adversary.

9.2 Implement syntax

We now implement the syntax of the formal model (see chapter 8) with our core protocol (see section 7.2). We are able to implement all functions as specified.

We explicitly note that both the formal model as well the core protocol assume the exact same roles with the same respective assumptions. Further, the terminology matches, too.

Setup The function consists out of Protocol 8 and Protocol 9. Besides the infrastructure assumptions, we assume some honest control component i, an honest setup authority and an honest printer authority.

The setup takes as an argument $params = (Id_1, \mathbb{V}, PtC)$.

The game is driven by honest instances which send each other messages over authentic and secure channels. The honest control component will only react when it has received the authentic message $(Id_1, |\mathbb{V}|, PtC)$ from the setup authority. The setup authority will only continue once it has received the partial ballots and shuffle public key over the secure channel of each control component. We therefore only need to analyse this single trace.

We provide the adversary with all data that is sent over authentic channels. Further, the adversary is able to contribute the values of the control components it controls.

At the end of the game, we explicitly define the initial secret state of voters and the honest control component, to formulate lemmas which help us proof the security properties. The voter's initial secret state consists out of the corresponding ballot sheet, which reaches the voter over secure channels via the trusted printer authority. The honest control components' initial secret state is the list of their partial ballots and their shuffle private key, which the honest control component generated itself.

Besides the secret state, the honest control component knows all values it needs to process the voting and tally phase ($\boldsymbol{hVA}, \boldsymbol{hCA}, \boldsymbol{CtE}$), all received authentically from the setup authority.

Algorithm: $\mathsf{Setup}^{\mathsf{i}}_{\mathcal{A}}(params)$
Input: Params: $params = (Id_1, \mathbb{V}, PtC)$

// the honest control component creates key material out of $(Id_1, |\mathbb{V}|, PtC)$
$\boldsymbol{Id} \leftarrow \mathsf{GenIds}(Id_1, |\mathbb{V}|)$
$\boldsymbol{b}^{(i)} \xleftarrow{r} \mathsf{GenBallots}(\boldsymbol{Id})$
$(sk_S^{(i)}, pk_S^{(i)}) \xleftarrow{r} \mathsf{SKGen}()$

// the adversary creates the key material of the other control components
$\{\boldsymbol{b}^{(j)} \mid j \in [1,m]\backslash i\} \leftarrow \mathcal{A}(Id_1, |\mathbb{V}|, PtC)$
$\{pk_S^{(j)} \mid j \in [1,m]\backslash i\} \leftarrow \mathcal{A}()$

// the setup authority combines and processes the key material
$\boldsymbol{B} \leftarrow \mathsf{MergeBallots}(\boldsymbol{Id}, \{\boldsymbol{b}^{(1)}, ..., \boldsymbol{b}^{(m)}\})$
$(\boldsymbol{hVA}, \boldsymbol{hCA}) \leftarrow \mathsf{HashAuths}(\boldsymbol{Id}, \boldsymbol{B})$
$\boldsymbol{BS} \leftarrow \mathsf{GenBallotSheets}(\boldsymbol{Id}, \boldsymbol{B}, \mathbb{V}, PtC)$
$pk_S \leftarrow \mathsf{SKAgg}(\{pk_S^{(1)}, ..., pk_S^{(m)}\})$
$\boldsymbol{CtE} \leftarrow \mathsf{GenCtEs}(\boldsymbol{Id}, \boldsymbol{B}, pk_S)$

// the setup authority publishes values for the voting and tally phases
$\{cc_j \mid j \in [1,m]\backslash i\} \leftarrow \mathcal{A}(\boldsymbol{hVA}, \boldsymbol{hCA}, pk_S, \boldsymbol{CtE})$

$\boldsymbol{v} \leftarrow \{BS_i \mid i \in [1, |\mathbb{V}|]\}$
$cc_i \leftarrow (\boldsymbol{b}^{(1)}, sk_S^{(1)})$

return $(\boldsymbol{v}, (cc_1, ..., cc_m))$

Algorithm 9.1: Setup under influence of the adversary $\mathcal{A}$, assuming honest control component i, honest setup authority and honest printer authority.

Lemma 9.1 (Secret voter state) *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, $\mathsf{Setup}^{i}_{\mathcal{A}}$ ensures that the values in each ballot sheet $BS_{Id} = (Id, VA, \boldsymbol{VV}, CA, CV, PtC_{Id}, \boldsymbol{R})$ (except Id) are unknown to the attacker. Concretely, the adversary cannot do better than guessing $(\boldsymbol{VV}, CV, \boldsymbol{R})$ uniform at random. The advantage of an adversary $\mathcal{B}$ guessing (VA, CA) is smaller or equal to $\boldsymbol{Adv}^{pIR}_{\mathcal{B},\mathsf{H}}$. The advantage of an adversary $\mathcal{D}$ guessing PtC_{Id} is smaller or equal to $\boldsymbol{Adv}^{IND\text{-}CPA}_{\mathcal{D},\mathsf{S}}$.*

Proof All operations (XOR, modular addition, chaining of permutations) used to combine the partial ballots in MergeBallots result in uniform at random values, as at least the partial value by the honest control component was picked uniform at random in GenBallots. GenBallotSheets, which generates the ballot sheets, preserves this property.

For the uniform at random VA and CA, the adversary additionally learns their hash values hVA and hCA. The advantage of an adversary learning VA given hVA is bound by the advantage of an adversary $\mathcal{B}$ against pre-image resistance of the hash function $\mathbf{Adv}^{\text{pIR}}_{\mathcal{B},\mathsf{H}}$ (see section 9.1). The same holds for the adversary learning CA given hCA.

For the uniform at random PtC_{Id} (as it is produced out of the uniform at random permutation P), the adversary learns the inversion CtE_{Id}. While the range of CtE_{Id} is encrypted, the adversary might be able to decrypt one of the $|CtE_{Id}| = |\mathbb{C}|$ ciphertexts. The advantage of that adversary is bound by $\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{D},\mathsf{S}}$.[4] □

Lemma 9.2 (Independent voter state) *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, $\mathsf{Setup}^i_{\mathcal{A}}$ ensures that the values inside $b^{(i)}_{Id}$ and BS_{Id} are independent for different voters. Concretely, even if the adversary is given other values produced during the same $\mathsf{Setup}^i_{\mathcal{A}}$ invocation, the adversary has no advantage than directly guessing the values (for which we bound the advantage in lemma 9.1).*

Proof At least the honest control component chooses in GenBallots independent values for each voter. The property therefore directly follows for $b^{(i)}_{Id}$. By the same argument as in the proof of lemma 9.1, the property also follows for all values in BS_{Id} under the bounded advantage.[5] □

Lemma 9.3 (Secret honest control component i state) *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, $\mathsf{Setup}^i_{\mathcal{A}}$ ensures that the values of the honest control component in each partial ballot $\boldsymbol{b}^{(i)}$ as well as the secret shuffle key $sk^{(i)}_S$ are unknown to the attacker. Concretely, the adversary cannot do better than guessing each value of each partial ballot $b = (va, \boldsymbol{vv}, ca, cv, p, \boldsymbol{r})$ uniform at random. Further, the advantage of an adversary $\mathcal{C}$ guessing $sk^{(i)}_S$, given pk_S, is bound by $\boldsymbol{Adv}^{KOW}_{\mathcal{C},\mathsf{S}}$.*

Proof The honest control component has chosen each value inside the partial ballots uniform at random, by the specification of GenBallots. These val-

[4] Note that the r used in each call to SEnc is uniform at random.

[5] We note that both pre-image resistance as well as IND-CPA security allow for these kind of oracles to exist: Given other voter's secret state, the adversary effectively learns some valid (CA, hCA) pairs and some valid decryptions of CtE.

ues are then never passed on to the adversary, as they are only transmitted over secure channels and processed by trusted authorities.[6]

The adversary advantage for guessing $sk_S^{(i)}$ follows directly from our infrastructure assumptions in section 9.1. □

Syntax to query protocol state We implement the functions to assess the state of the voter as follows:

- $\mathsf{Confirmed}(\mathrm{V})$ returns true if the voter V has already published their confirmation authentication, otherwise false.
- $\mathsf{IVCheck}(\mathrm{V})$ returns true if the voter V has successfully compared the confirmation verification codes, otherwise false.[7]

We implement the functions over the bulletin board as follows:

- $\mathsf{Count}(\mathrm{BB})$ returns $|\boldsymbol{\mathcal{C}}_{Conf}|$.
- $\mathsf{Extract}(\mathrm{BB}, i)$ retrieves $\mathcal{C} \xleftarrow{} (Id_i, .) \in \boldsymbol{\mathcal{C}}_{Conf}$. If $\mathcal{C} = \perp$, it returns $\perp$. Else, it retrieves $\mathcal{P} \xleftarrow{} (., \mathcal{C}) \in PtC_{Id_i}$. $\boldsymbol{\mathcal{C}}_{Conf}$ never contains more than one vote per Id, as before an entry with Id is added to $\boldsymbol{\mathcal{C}}_{Conf}$, it is ensured that $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf}$ (see Protocol 11).
- $\mathsf{Validate}(\mathrm{BB}, i)$ returns true if $\mathsf{Extract}(\mathrm{BB}, i) \neq \perp$, otherwise false.[8]
- $\mathsf{CorrectTally}(\mathrm{BB})$ returns the sum of all plain votes $\boldsymbol{\mathcal{P}} = \{\mathcal{P} \mid (\mathcal{P}, \mathcal{C}) \in PtC_{Id} \mid (Id, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf} \mid Id \in \boldsymbol{Id}\}$.
- $\mathsf{VerifyTally}(\mathrm{BB}, t)$ executes $\boldsymbol{\mathcal{E}}_0 = \mathsf{MapCtEs}(\boldsymbol{\mathcal{C}}_{Conf}, \boldsymbol{CtE})$ and then $\forall i \in [1, m].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{i-1}, \boldsymbol{\mathcal{E}}_i, \pi_i)$ for $t = \{(\boldsymbol{\mathcal{E}}_1, \pi_1), ..., (\boldsymbol{\mathcal{E}}_m, \pi_m)\}$. If this is successful, it returns true, otherwise false.

In our proof, there is no bulletin board, rather the honest control component takes over its responsibilities. When we need state of the bulletin board, we therefore use the view the honest control component has of it.[9]

Tally The function consists out of Protocol 12 and Protocol 13. We assume some honest control component i.

The tally takes as an argument the confirmed votes, the $\boldsymbol{CtE}$ lookup and the shuffle secret key of the honest control component.

The game is driven by the honest control component sending and receiving values over authentic channels from the other control components. It expects

[6] At this point of the game, there are no compromised voters, hence this statement holds.

[7] Note that for honest voters, $\mathsf{IVCheck}(\mathrm{V}) \implies \mathsf{Confirmed}(\mathrm{V})$.

[8] The bulletin board, by construction, does not accept invalid votes.

[9] Note that the tally phase guarantees the $\boldsymbol{\mathcal{C}}_{Conf}$ as seen by the honest control component is used for tallying.

the result of the shuffles of the previous control components, and verifies their proofs. Then, the honest control component continues with its own shuffle decryption. It then expects the result of the shuffles of the subsequent control components, and again verifies their proofs. We therefore only need to analyse this single trace.

The adversary receives the shuffle result and proof of the honest control component. Further, the adversary is able to contribute the shuffle results and proofs of the control components it controls. Note that adversary knows $\boldsymbol{\mathcal{C}}_{Conf}$ and $\boldsymbol{CtE}$ from the setup and the voting phase, which is why we do not explicitly pass it to the adversary in Algorithm 9.2.

The honest control component will abort whenever some proof verification fails. Only if all control components reach the end of their tally phase (hence have verified the proofs of all other control components, and have executed their own shuffle), will the tally result be announced.

Algorithm: $\mathsf{Tally}^{\mathsf{i}}_{\mathcal{A}}(\boldsymbol{\mathcal{C}}_{Conf}, \boldsymbol{CtE}, sk_S^{(i)})$

$\boldsymbol{\mathcal{E}}_0 \leftarrow \mathsf{MapCtEs}(\boldsymbol{\mathcal{C}}_{Conf}, \boldsymbol{CtE})$

// the adversary provides previous shuffles
$\{(\boldsymbol{\mathcal{E}}_1, \pi_1), ..., (\boldsymbol{\mathcal{E}}_{i-1}, \pi_{i-1})\} \leftarrow \mathcal{A}()$

// the honest control component i verifies previous shuffles and shuffles
$\forall j \in [1, i-1].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{j-1}, \boldsymbol{\mathcal{E}}_j, \pi_j)$

$(\boldsymbol{\mathcal{E}}_i, \pi_i) \xleftarrow{r} \mathsf{SShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{i-1}, sk_S^{(i)})$

// the adversary provides subsequent shuffles
$\{(\boldsymbol{\mathcal{E}}_{i+1}, \pi_{i+1}), ..., (\boldsymbol{\mathcal{E}}_m, \pi_m)\} \leftarrow \mathcal{A}(\boldsymbol{\mathcal{E}}_i, \pi_i)$

// the honest control component i verifies subsequent shuffles
$\forall j \in [i+1, m].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{j-1}, \boldsymbol{\mathcal{E}}_j, \pi_j)$

return $(\{(\boldsymbol{\mathcal{E}}_1, \pi_1), ..., (\boldsymbol{\mathcal{E}}_m, \pi_m)\})$

Algorithm 9.2: Tally under influence of the adversary $\mathcal{A}$.

Lemma 9.4 (Correct tally) *Under the infrastructure assumptions, for honest control component i, $\mathsf{Tally}^{i}_{\mathcal{A}}$ is correct, hence it outputs exactly what is represented by $\boldsymbol{\mathcal{C}}_{Conf}$ and $\boldsymbol{CtE}$. Concretely, the advantage of an adversary $\mathcal{G}$ convincing the honest control component of a different tally result is bound by $\boldsymbol{Adv}^{verify}_{\mathcal{G},S}$.*

Proof The honest control component i derives $\boldsymbol{\mathcal{E}}_0$ directly from $\boldsymbol{\mathcal{C}}_{Conf}$ and $\boldsymbol{CtE}$. Using SVerifyShuffleDecrypt, it validates proofs of all other control components which ensure each pair of ciphertext $\forall j \in [1, m] \backslash i.(\boldsymbol{\mathcal{E}}_{i-1}, \boldsymbol{\mathcal{E}}_i)$

represent the same plaintext. Further, as SVerifyShuffleDecrypt is executed honestly and therefore correct, the two lists $\boldsymbol{\mathcal{E}}_{i-1}$ and $\boldsymbol{\mathcal{E}}_i$ represent the same plaintext, too. It follows that the final list $\boldsymbol{\mathcal{E}}_m$ represents the same as $\boldsymbol{\mathcal{C}}_{Conf}$ and $\boldsymbol{CtE}$.

The advantage of an adversary breaking the correctness of one of the $m-1$ honestly executed calls to SVerifyShuffleDecrypt is bound by $\mathbf{Adv}^{\text{verify}}_{\mathcal{G},\mathsf{S}}$. □

Lemma 9.5 (Private tally) *Under the infrastructure assumptions, for honest control component i, $\mathsf{Tally}^i_{\mathcal{A}}$ is private, hence does not reveal anything but the set of plain votes represented by represented by $\boldsymbol{\mathcal{C}}_{Conf}$ and $\boldsymbol{CtE}$. Concretely, the advantage of an adversary learning anything more about an individual voter's preference than what is revealed by the tally result itself is bound by $\boldsymbol{Adv}^{IND\text{-}CPA}_{\mathcal{E},\mathsf{S}}$, $\boldsymbol{Adv}^{shuffle}_{\mathcal{F},\mathsf{S}}$ and $\boldsymbol{Adv}^{verify}_{\mathcal{G},\mathsf{S}}$.*

Proof We first argue that the adversary cannot change the plaintext represented by $\boldsymbol{\mathcal{E}}_{i-1}$ - compared to the plaintext of the correct tally - by a similar argumentation as done in the proof for *lemma* 9.4: The adversary would need to break the correctness of at least one of the $i-1$ invocations of SVerifyShuffleDecrypt executed by the honest control component, for which the advantage of the adversary is bound by $\mathbf{Adv}^{\text{verify}}_{\mathcal{G},\mathsf{S}}$.

When the honest control component then executes SShuffleDecrypt using $\boldsymbol{\mathcal{E}}_{i-1}$, it produces $\boldsymbol{\mathcal{E}}_i$ such that the adversary advantage to learn which $\mathcal{E} \in \boldsymbol{\mathcal{E}}_{i-1}$ corresponds to which $\mathcal{E}' \in \boldsymbol{\mathcal{E}}_i$ is bound by $\mathbf{Adv}^{\text{shuffle}}_{\mathcal{F},\mathsf{S}}$.

Further, the adversary advantage to learn the plaintext directly from any of the $|\boldsymbol{\mathcal{C}}_{Conf}| * i$ ciphertext in $\boldsymbol{\mathcal{E}}_i$ for $i \in [0, i-1]$ is bound by $\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{E},\mathsf{S}}$.

We summarize that as soon as the honest control component has executed its shuffle, the votes cannot be attributed any more to individual voters. The adversary cannot change the input of the honest shuffling, and cannot directly decrypt the votes before the honest control component shuffles. □

9.3 Security properties

Given the implementation of the syntax of the formal model, we now use its game-based definitions of the security properties.

Fixing the honest voter and control component We argue that fixing the honest voter to V_1 and the honest control component to CC_1 does not incur a loss in generality.

We do this with a simple argument: We observe that all control components as well as all voters have the same specification, hence honest instances will

all behave the same. The adversary therefore cannot have an advantage depending which instance is honest (even if it has interacted with the protocol first).

Oracles in the voting phase To avoid duplicating specification of section 7.2, we do not explicitly define oracles provided by the voters and control components in the voting phase. We however will refer to the oracles in the following text, hence we clear up terminology.

Honest control components provide two oracles: The *Cast Oracle* which allows to cast a vote (the first round-trip with the voter), and the *Confirm Oracle* which allows to confirm a vote (the second round-trip with the voter). The voter provides three oracles which remain unnamed: One to cast a vote, one to confirm a vote, and one to process the final confirm verification.

9.3.1 Individual Verifiability

We instantiate and give bounds of the individual verifiability properties as defined in section 8.2.1.

9.3.1.1 Individual Verifiability

The adversary wins the game if the individual verifiability check of the honest voter passes, although there is no confirmed vote of that voter, or the confirmed vote does not represent the honest voter's selection.

In our game, we restrict ourselves to proof the advantage of an adversary against a single trace in the voting phase. We however argue this does not incur a loss of generality, as the adversary's advantage cannot increase for any other trace.

For the individual verifiability check of the honest voter to succeed the honest voter must have executed their specification in order and until the end. We therefore let the game be driven by the honest voter. Any message sent by the voter is given to the adversary, and any message received by the voter is provided by the adversary. When the message received back from the adversary is wrong, the game aborts: As the voter's individual verifiability check will not succeed, the adversary cannot win the game.

To derive the messages for the honest voter, the adversary might query other instances. We argue that by interacting with dishonest control components and dishonest voters, the adversary gets no advantage: Their state is already under the control of the adversary, hence the adversary does not learn anything new. Further, it does not influence the state of the honest voter and honest control component, and therefore it does not influence whether the property is preserved or broken. Hence the only remaining oracles which

potentially have an advantage to the adversary are provided by the honest control component.[10]

Queries to the honest control component The adversary is able to interact the cast oracle with (Id, VA, C) and the confirm oracle with (Id, CA). We argue that for both oracles, the adversary gets no advantage when querying with $Id' \neq Id$, as then state corresponding to the honest voter is unmodified (hence, whether the property holds or not is unchanged). Further, the adversary does not receive any useful answer, as with $Id' \neq Id$ any answer is independent to the state corresponding to the voter b_{Id} (see lemma 9.2). We therefore fix the adversary to always query with Id provided by the honest voter.

If the adversary wants to query the cast oracle, we force it to do so before responding to the voter with the vote verification. Calling it later in the game cannot increase the adversaries advantage to guess the right vote verification, but would still have the same effect on the honest control components state. Calling it earlier would make it harder to pass validation (for example, the adversary needs to guess some VA' such that $\mathsf{Hash}(VA') = hCA$), but when validation is not passed, the oracle has no effect (no state changed or revealed).

Similarly, if the adversary wants to query the confirm oracle, we force it to do so before responding to the voter with the confirm verification. Calling it later cannot increase the adversaries advantage to guess the right confirm verification, calling it earlier cannot increase the adversary advantage to pass validation.

At this point, we can also restrict the adversary to call the same oracle at most once. By the time the adversary calls the oracle, it knows how it can pass validation (hence it does not need multiple attempts). Further, after the first call passed validation, any subsequent oracle call will not modify state or return new values any more.

As it is now predetermined when the adversary calls which oracle, we can inline the oracles directly into the game. As it makes our game description easier, we do not explicitly model an adversary not calling an oracle: If the adversary does not wish to change the honest control components state it can simply forward invalid authentication.[11]

Deriving the game We derive our game from Game 1 of the formal model as follows:

[10] The honest voter's oracles are all already in use.

[11] Alternatively, we could let the adversary explicitly return a value whether it wants to call the oracle or not, and remove its option to specify VA'. It matters not for our proof.

- We replace $\mathsf{Setup}_{\mathcal{A}}$ with our implemented $\mathsf{Setup}^1_{\mathcal{A}}$ (and inherit its trust assumptions).
- After the adversary specifies the voting preference of the honest voter, we inline the specification of the voter of Protocol 10 and Protocol 11. Any sent message is passed to the adversary, and any received message is received from the adversary. If some received message is wrong, the game is aborted (and the adversary loses).
- We inline the cast oracle of the honest control component after the cast message is sent by the voter, giving the adversary the power to specify VA', C' and possibly learning $vv^{(1)}_{C'}$. Further, we inline the confirm oracle of the honest control component after the confirm message is sent by the voter, giving the adversary the power to specify CA' and possibly learning $cv^{(1)}$.
- We make the attacker slightly more powerful by passing it the private state of the dishonest voters and control components already before it specifies the voting preference of the honest voter.
- We replace the syntax of the assertion with our implemented syntax. We remove the left-hand side and directly assert the implication, as the left-hard side is implied if the game proceeded that far.

This results in Game 9.

The adversary wins the game if it runs until the end, but $\mathcal{P}' \neq \mathcal{P}$. We define the advantage of an adversary against individual verifiability as:

$$\mathbf{Adv}^{\mathrm{IV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{IV}}_{\mathcal{A}} \text{ assert fails}]$$

The following theorem provides a bound on the advantage of any (even unbounded) adversary.

Theorem 9.6 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{IV}_{\mathcal{A}} \leq \frac{1}{|\mathbb{Z}_v|} + \frac{1}{|\mathbb{Z}_v|}$$

Intuitively, the first part is from guessing the vote verification, and the second part from guessing the confirmation verification. The adversary is unbounded as no computational assumptions are required to safeguard the verification codes. The proof for the theorem is in section 9.4.1.

$\mathbf{Exp}^{\mathrm{IV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

// the state of the honest voter 1
$(Id, VA, \{VV_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, CA, CV, PtC_{Id}) \leftarrow v_1$

// the attacker receives the private state of the dishonest voters and control components and is able to pick the honest voter's plain vote
$\mathcal{P} \leftarrow \mathcal{A}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

// the voter picks the corresponding $\mathcal{C}$ and casts the vote
$\mathcal{C} \xleftarrow{\cdot} (\mathcal{P}, .) \in PtC_{Id}$

// the attacker decides what to forward to the honest control component
$(VA', \mathcal{C}') \leftarrow \mathcal{A}(VA, \mathcal{C})$

// the honest control component stores the cast vote and answers
if $((Id, \mathsf{Hash}(VA')) \in \boldsymbol{hVA}$ and $\mathcal{C}' \in \mathbb{C}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Cast})$
then $\boldsymbol{\mathcal{C}}_{Cast} \leftarrow \boldsymbol{\mathcal{C}}_{Cast} \cup \{(Id, \mathcal{C}')\}$; $(..., \{vv^{(1)}_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, ...) \leftarrow b^{(1)}_{Id}$; $\mathcal{A}(vv^{(1)}_{\mathcal{C}'})$

// the attacker decides what to forward to the voter
$(\{vv'^{(1)}_{\mathcal{C}}, ..., vv'^{(m)}_{\mathcal{C}}\}) \leftarrow \mathcal{A}()$

// the voter checks the vote verification
if $(VV_{\mathcal{C}} \neq \oplus^{m}_{i=1} vv'^{(i)}_{\mathcal{C}})$ then abort

// the voter confirms the vote
$\mathcal{A}(CA)$

// the attacker decides what to forward to the honest control component
$(CA') \leftarrow \mathcal{A}()$

// the honest control component stores the confirmed vote and answers
$\mathcal{C}'' \leftarrow (Id, .) \in \boldsymbol{\mathcal{C}}_{Cast}$
if $((Id, \mathsf{Hash}(CA')) \in \boldsymbol{hCA}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf})$
then $\boldsymbol{\mathcal{C}}_{Conf} \leftarrow \boldsymbol{\mathcal{C}}_{Conf} \cup \{(Id, \mathcal{C}'')\}$; $(..., cv^{(1)}, ...) \leftarrow b^{(1)}_{Id}$; $\mathcal{A}(cv^{(1)})$

// the attacker decides what to forward to the voter
$(\{cv'^{(1)}, ..., cv'^{(m)}\}) \leftarrow \mathcal{A}()$

// the voter checks the vote verification
if $(CV \neq \oplus^{m}_{i=1} cv'^{(i)})$ then abort

// the attacker wins if $\mathcal{P}' \neq \mathcal{P}$
$\mathcal{C}''' \xleftarrow{\cdot} (Id, .) \in \boldsymbol{\mathcal{C}}_{Conf}$
$\mathcal{P}' \xleftarrow{\cdot} (., \mathcal{C}''') \in PtC_{Id}$
assert $\mathcal{P}' = \mathcal{P}$

Game 9: Game for individual verifiability. We let the adversary choose $\mathcal{P}$, the plain vote of the voter. Note that $\mathcal{C}'''$ as well as $\mathcal{P}'$ might be undefined.

9.3.1.2 Participation Verifiability

Participation verifiability allows voters to verify whether they have participated in the election or not. As proposed in section 8.2.1 we split the definition into $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$ for participating voters, and $\mathbf{Exp}^{\text{PVA}}_{\mathcal{A}}$ for abstaining voters. We proof $\mathbf{Adv}^{\text{PVA}}_{\mathcal{A}} = \mathbf{Adv}^{\text{EV}}_{\mathcal{A}}$ in section 9.3.4.1. Here, we only discuss $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$.

We can easily construct $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$ out of $\mathbf{Exp}^{\text{IV}}_{\mathcal{A}}$ by weakening the assert statement: We only need to assert that $(Id, .) \in \boldsymbol{\mathcal{C}}_{Conf}$, hence that *some* vote is confirmed. We avoid describing the full game again.

The following theorem provides a bound on the advantage of any (even unbounded) adversary against participation verifiability - participating voters.

Theorem 9.7 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{PVP}_{\mathcal{A}} \leq \frac{1}{|\mathbb{Z}_v|}$$

Intuitively, the advantage is from guessing the confirmation verification. The proof is in section 9.4.2

From theorem 9.7 and theorem 9.13 we can directly bound the advantage of an adversary against participation verifiability.

Theorem 9.8 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any polynomially bound adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{PV}_{\mathcal{A}} \leq \frac{1}{|\mathbb{Z}_v|} + \boldsymbol{Adv}^{pIR}_{\mathcal{B},\mathsf{H}}$$

Proof We know that $\mathbf{Adv}^{\text{PV}}_{\mathcal{A}} \leq \mathbf{Adv}^{\text{PVP}}_{\mathcal{A}} + \mathbf{Adv}^{\text{PVA}}_{\mathcal{A}}$ and that $\mathbf{Adv}^{\text{PVA}}_{\mathcal{A}} = \mathbf{Adv}^{\text{EV}}_{\mathcal{A}}$ (see section 8.2.1). By theorem 9.7, it holds that $\mathbf{Adv}^{\text{PVP}}_{\mathcal{A}} \leq \frac{1}{|\mathbb{Z}_v|}$. By theorem 9.13 it holds that $\mathbf{Adv}^{\text{EV}}_{\mathcal{A}} \leq \mathbf{Adv}^{\text{pIR}}_{\mathcal{B},\mathsf{H}}$. □

9.3.1.3 Vote Verifiability

Vote verifiability ensures all votes represent valid voting options.

Vote verifiability is guaranteed by the cast oracle of the honest control component: Before some $\mathcal{C}$ is added to $\boldsymbol{\mathcal{C}}_{Cast}$, it is ensured that $\mathcal{C} \in \mathbb{C}$. By definition, $\mathcal{C}$ is then a valid vote.

Theorem 9.9 *Under the infrastructure assumptions, for honest setup authority and honest control component i, for any adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}_{\mathcal{A}}^{VV} = 0$$

Proof We observe that in $\mathsf{Setup}_{\mathcal{A}}^{i}$ the setup authority delivers the honest control component *PtC*, which can then be used to derive $\mathbb{C}$. Given an honest setup authority, we are ensured the honest control component indeed received the correct $\mathbb{C}$.

We then use a proof by contradiction. Assuming for some voter index i, Extract returns some vote $\mathcal{C} \notin \mathbb{C}$. This means that $(Id_i, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf}$ (by definition of Extract). Any $(Id_i, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf}$ must also be in $\boldsymbol{\mathcal{C}}_{Cast}$ (by Protocol 11). Any $(Id_i, \mathcal{C})$ added to $\boldsymbol{\mathcal{C}}_{Cast}$ is assured to fulfil $\mathcal{C} \in \mathbb{C}$ (by Protocol 10). This contradicts our initial assumption, and concludes the proof. □

9.3.2 Universal Verifiability

We instantiate and give bounds of the universal verifiability property as defined in section 8.2.2.

9.3.2.1 Universal Verifiability

Universal verifiability ensures the confirmed votes are tallied correctly.

For universal verifiability, we assume an even stronger adversary: We no longer rely on any trusted parties except the honest control component i.

We cannot reuse $\mathsf{Setup}_{\mathcal{A}}^{i}$ as done in the previous games, as it relies on a trusted setup and printer authority. We therefore redesign it here under the new trust assumption that only the single honest control component exists. As before, the honest control component receives $params = (Id_1, \mathbb{V}, PtC)$. Based on this, the honest control component generates its key material. The adversary then learns the partial ballots (as it is sent to the setup authority), and freely decides upon the result of the setup function.

In the voting phase, the adversary essentially plays only against the honest control component, casting and confirming arbitrary votes. However, here the capabilities of this otherwise very strong adversary are heavily constrained: While for universal verifiability the adversary can do more or less everything, the other properties still hold under the weaker adversary model.[12] Concretely, after the voting phase it must hold that $\boldsymbol{\mathcal{P}} = \{\mathcal{P} \mid (\mathcal{P}, \mathcal{C}) \in PtC_{Id} \mid (Id, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf} \mid Id \in \boldsymbol{Id}\}$ is the real tally result. By eligibility uniqueness (see section 9.3.4.2), for each Id at most one $\mathcal{C}$ will be selected. Then, by individual verifiability (see section 9.3.1.1), it must hold that this

[12] Like the Genie in Aladdin: *PHENOMINAL COSMIC POWER...itty bitty living space.*

$\mathcal{C}$ reflects the $\mathcal{P}$ of the corresponding voter, as does the corresponding entry in CtE_{Id}.

After the voting phase $\mathsf{Tally}^i_{\mathcal{A}}$ is run, which establishes the tally result t. The adversary wins if this tally result passes verification, although does not represent the real tally result.[13]

Deriving the game We derive our game from Game 4 of the formal model as follows:

- We inline the new setup function as described.
- We construct the honest control component, and pass it as an oracle to the adversary.
- We replace $\mathsf{Tally}_{\mathcal{A}}$ with our implemented $\mathsf{Tally}^i_{\mathcal{A}}$.
- We replace the syntax of the assertion with our implemented syntax. We abort if the left-hand side is false. Afterwards, we assert the implication.

This results in Game 9.

The adversary wins the game if it runs until the end, but the assertion fails. We define the advantage of an adversary against universal verifiability as:

$$\mathbf{Adv}^{\mathrm{IV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{IV}}_{\mathcal{A}} \text{ assert fails}]$$

The following theorem provides a bound on the advantage of any polynomially bounded adversary.

Theorem 9.10 *Under the infrastructure assumptions, for honest control component i, for any polynomially bound adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{UV}_{\mathcal{A}} \leq \boldsymbol{Adv}^{verify}_{\mathcal{G},\mathsf{S}}$$

when eligibility uniqueness (see section 9.3.4.2) and individual verifiability (see section 9.3.1.1) also hold.

Proof By eligibility uniqueness (see section 9.3.4.2) and individual verifiability (see section 9.3.1.1), we know that right before $\mathsf{Tally}^i_{\mathcal{A}}$ is executed, it holds that $\boldsymbol{\mathcal{P}} = \{\mathcal{P} \mid (\mathcal{P}, \mathcal{C}) \in PtC_{Id} \mid (Id, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf} \mid Id \in \boldsymbol{Id}\}$.[14]

The advantage of the adversary being able to influence $\mathsf{Tally}^i_{\mathcal{A}}$ such that $\boldsymbol{\mathcal{E}}_m \neq \boldsymbol{\mathcal{P}}$ directly follows from the correctness property of the verifiable shuffle (see lemma 9.4). □

[13] We explicitly include the verification in our game, although at this point it will always pass. If it would not, the honest control component would have aborted during Tally.

[14] See the discussion from above we avoid to duplicate.

$\mathbf{Exp}^{\mathrm{UV}}_{\mathcal{A}}(params)$:

// the honest control component creates key material out of $(Id_1, |\mathbb{V}|, PtC)$
$\boldsymbol{Id} \leftarrow \mathsf{GenIds}(Id_1, |\mathbb{V}|)$
$\boldsymbol{b}^{(i)} \xleftarrow{r} \mathsf{GenBallots}(\boldsymbol{Id})$
$(sk_S^{(i)}, pk_S^{(i)}) \xleftarrow{r} \mathsf{SKGen}()$

// the adversary decides the result of the setup phase
$(\boldsymbol{hVA}, \boldsymbol{hCA}, \boldsymbol{CtE}, pk_S) \leftarrow \mathcal{A}(\boldsymbol{b}^{(i)})$

// the adversary plays the voting phase with the honest control component
$cc_i \leftarrow (\boldsymbol{b}^{(1)}, sk_S^{(1)})$
$\mathrm{CC}_i \xleftarrow{c} \mathbf{CC}(cc_i)$
$\mathcal{A}^{\mathrm{CC}_i}()$

// the tally phase is run
$t \leftarrow \mathsf{Tally}^i_{\mathcal{A}}()$

// if the tally result verifies...
$\{(\boldsymbol{\mathcal{E}}_1, \pi_1), \ldots, (\boldsymbol{\mathcal{E}}_m, \pi_m)\} = t$
$\boldsymbol{\mathcal{E}}_0 = \mathsf{MapCtEs}(\boldsymbol{\mathcal{C}}_{Conf}, \boldsymbol{CtE})$
if not $\forall i \in [1, m].\mathsf{SVerifyShuffleDecrypt}(\boldsymbol{\mathcal{E}}_{i-1}, \boldsymbol{\mathcal{E}}_i, \pi_i)$ then abort

// ...then it must be correct
assert $\boldsymbol{\mathcal{P}} = \{\mathcal{P} \mid (\mathcal{P}, \mathcal{C}) \in PtC_{Id} \mid (Id, \mathcal{C}) \in \boldsymbol{\mathcal{C}}_{Conf} \mid Id \in \boldsymbol{Id}\} = \boldsymbol{\mathcal{E}}_m$

Game 10: Game for universal verifiability for some honest control component i.

9.3.3 Vote Secrecy and Fairness

We instantiate and give bounds of the vote secrecy and fairness properties as defined in section 8.2.3.

9.3.3.1 Vote Secrecy

Vote secrecy ensures the adversary does not learn the voter selection.

To win the game, the adversary needs to distinguish whether it plays in the real world, with challenge bit $\beta = 0$, or the fake world, with challenge bit $\beta = 1$. The adversary is granted the ability to decide upon the selection the honest voter casts in the real world $\mathcal{P}_0$ or in the fake world $\mathcal{P}_1$.

The definition additionally requires $\mathsf{SimTally}_{\mathcal{A}}$, which we implement with $\mathsf{SimTally}^i_{\mathcal{A}}$. $\mathsf{SimTally}^i_{\mathcal{A}}$ behaves exactly the same as $\mathsf{Tally}^i_{\mathcal{A}}$, but the call to SShuffleDecrypt is replaced by a call to SSimShuffleDecrypt. This function behaves exactly the same as SShuffleDecrypt, but it replaces the selection

the honest voter cast in the fake world by the selection the honest voter would have cast in the real world.[15] We assume that the advantage of an adversary distinguishing Tally from SimTally is bounded by the advantage of the adversary against the secrecy of the tally (see lemma 9.5).[16]

The challenger bit β influences the game in two cases: First, in the voting phase, it determines which attacker-chosen voter selection $(\mathcal{P}_0, \mathcal{P}_1)$ the honest voter actually casts as $\mathcal{C} = PtC_{Id} * \mathcal{P}_\beta$. Second, in the tally phase, it determines whether Tally or SimTally is executed.

To predict β in the voting phase, the adversary's only chance to distinguish game 0 from game 1 is to correctly predict whether $\mathcal{C}$ represents $\mathcal{P}_0$ or $\mathcal{P}_1$. We argue that no oracle in the voting phase will increase the adversaries advantage. The honest voter will only additionally reveal authentication values or whether verification values were correct, which both are unrelated to $\mathcal{C}$ or $\mathcal{P}_\beta$. The honest control component will only reveal verification codes or whether authentication values were correct, which both are unrelated to $\mathcal{C}$ or $\mathcal{P}_\beta$. We therefore omit providing the adversary with oracles in the voting phase.

Deriving the game We derive our game from Game 5 of the formal model as follows:

- We replace $\mathsf{Setup}_\mathcal{A}$ with our implemented $\mathsf{Setup}^1_\mathcal{A}$ (and inherit its trust assumptions).
- We inline the cast oracle of the honest voter. We let the adversary specify $\mathcal{P}_0$ and $\mathcal{P}_1$ and pass it the $\mathcal{C}$ the voter casts.
- We replace $\mathsf{Tally}_\mathcal{A}$ with our implemented $\mathsf{Tally}^1_\mathcal{A}$.
- We replace $\mathsf{SimTally}_\mathcal{A}$ with our implemented $\mathsf{SimTally}^1_\mathcal{A}$.
- We make the attacker slightly more powerful by passing it the private state of the dishonest voters and control components already before it specifies the voting preferences of the honest voter.

The adversary wins the game if it can distinguish $\mathbf{Exp}^{\mathrm{VS},0}_\mathcal{A}(params)$ from $\mathbf{Exp}^{\mathrm{VS},1}_\mathcal{A}(params)$. We define the advantage of an adversary against vote secrecy as:

$$\mathbf{Adv}^{\mathrm{VS}}_\mathcal{A} = |\mathbf{Exp}^{\mathrm{VS},0}_\mathcal{A}(params) - \mathbf{Exp}^{\mathrm{VS},1}_\mathcal{A}(params)|$$

The following theorem provides a bound on the advantage of any polynomially bounded adversary.

[15] This essentially ensures that the tally result always reflects the tally of the real world.

[16] Whether this really holds in the verifiable shuffle our infrastructure then really provides needs to be proven, and remains here an educated guess.

$\mathbf{Exp}_{\mathcal{A}}^{\mathrm{VS},\beta}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

// the state of the honest voter 1
$(Id, VA, \{VV_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, CA, CV, PtC_{Id}) \leftarrow v_1$

// the attacker receives the private state of the dishonest voters and control components and is able to pick the honest voter's plain vote
$(\mathcal{P}_0, \mathcal{P}_1) \leftarrow \mathcal{A}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

// the voter picks the corresponding $\mathcal{C}$ and casts the vote
$\mathcal{C} \xleftarrow{\cdot} (\mathcal{P}_\beta, .) \in PtC_{Id}$

// the attacker learns $\mathcal{C}$
$\mathcal{A}(\mathcal{C})$

// the tally is executed
if $\beta = 0$ then $r \leftarrow \mathsf{Tally}_{\mathcal{A}}()$ else $r \leftarrow \mathsf{SimTally}_{\mathcal{A}}()$

// the adversary guesses the challenge bit
$d \leftarrow \mathcal{A}(r)$
if $d = \beta$ then return 1 else return 0

Game 11: Game for vote secrecy. We let the adversary choose $\mathcal{P}_0$ and $\mathcal{P}_1$, of which the voter will cast $\mathcal{P}_\beta$.

Theorem 9.11 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any polynomially bounded adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}_{\mathcal{A}}^{VS} \leq \boldsymbol{Adv}_{\mathcal{E},\mathsf{S}}^{IND\text{-}CPA} + \boldsymbol{Adv}_{\mathcal{F},\mathsf{S}}^{shuffle} + \boldsymbol{Adv}_{\mathcal{G},\mathsf{S}}^{verify}$$

Proof The adversary does not know PtC_{Id} which is a uniform at random permutation (see lemma 9.1 and lemma 9.2). Therefore the advantage of an adversary detecting whether $\mathcal{P}_0$ or $\mathcal{P}_1$ was used is 0.[17]

The adversary might be able to tell apart $\mathsf{Tally}_{\mathcal{A}}^1$ and $\mathsf{SimTally}_{\mathcal{A}}^1$. As assumed, the advantage of an adversary distinguishing these two functions is bounded by the advantage of the adversary against the secrecy of the tally (see lemma 9.5).

Collecting all advantages proofs the theorem. □

[17] Strictly speaking, this ignores that the adversary knows CtE_{Id}, and the ciphertext $\mathcal{E}$ associated to $\mathcal{C}$ might also give the adversary some advantage. However, the privacy of the tally bound already includes how the adversary might attempt decryption of $\mathcal{E}$, which is why we do not include it here again.

9.3.3.2 Fairness

As noted in section 8.2.3, fairness is a weaker property than vote secrecy. We can therefore bound the advantage of an adversary against fairness by the advantage of an attacker against vote secrecy.

Theorem 9.12 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any polynomially bounded adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{F}_{\mathcal{A}} \leq \boldsymbol{Adv}^{VS}_{\mathcal{A}}$$

9.3.4 Authentication

We instantiate and give bounds of the authentication properties as defined in section 8.2.4.

9.3.4.1 Eligibility Verifiability

Eligibility verifiability ensures every vote considered in the tally is confirmed by an eligible voter.

The adversary wins the game if the honest voter has not confirmed their vote, but their vote is nonetheless marked as confirmed. In our game, we restrict ourselves to proof the advantage of an adversary against a single trace in the voting phase. We however argue this does not incur a loss of generality, as the adversary's advantage cannot increase for any other trace.

For the adversary to win, it is required that the vote of the honest voter is stored as confirmed at the honest control component. This necessarily requires the adversary to call the confirm oracle and pass validation, hence guess the correct request (Id, CA). The adversary knows Id (as it is public), but does not know CA after the setup (see lemma 9.1). To learn more about CA the adversary might query other instances.

As in section 9.3.1.1, we observe that querying instances the adversary already has control over results in no advantage to the adversary. Querying the honest control component to learn CA results in no advantage, as the honest control component never reveals authentication secrets (like the partial va). Querying the honest voter for the confirmation (the only way the voter reveals CA) would result in a trivial lose for the adversary, as then it can never break the assertion. We conclude that the adversary cannot do better through queries to other instances.

Deriving the game We derive our game from Game 7 of the formal model as follows:

- We replace $\mathsf{Setup}_{\mathcal{A}}$ with our implemented $\mathsf{Setup}^1_{\mathcal{A}}$ (and inherit its trust assumptions).
- We inline the confirm oracle of the honest control component, and let the adversary specify CA'.
- We remove that the adversary specifies the vote preference of the honest voter, as the honest voter never casts a vote. We also remove how the honest control component answers if the confirmation was successful, as the adversary is not active any more afterwards.
- We replace the syntax of the assertion with our implemented syntax. We remove the left-hand side and directly assert the implication, as the left-hard side is implied: We do not allow the adversary to query the voter in such a way that they would confirm their vote.

$\mathbf{Exp}^{\mathrm{EV}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

```
// the id of the honest voter 1
```
$(Id,\ldots) \leftarrow v_1$

```
// the attacker receives the private state of the dishonest voters and
control components
```
$\mathcal{A}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

```
// the attacker decides how to query the honest control component
```
$(CA) \leftarrow \mathcal{A}()$

```
// the honest control component stores the confirmed vote
```
$\mathcal{C} \leftarrow (Id, .) \in \boldsymbol{\mathcal{C}}_{Cast}$
if $((Id, \mathsf{Hash}(CA)) \in \boldsymbol{hCA}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf})$
then $\boldsymbol{\mathcal{C}}_{Conf} \leftarrow \boldsymbol{\mathcal{C}}_{Conf} \cup \{(Id, \mathcal{C})\}$

```
// the attacker wins if some vote is confirmed
```
assert $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf}$

Game 12: Game for eligibility verifiability. We let the adversary choose $\mathcal{P}$, the plain vote of the voter.

The adversary wins the game if the vote is confirmed in the end. We define the advantage of an adversary against eligibility verifiability as:

$$\mathbf{Adv}^{\mathrm{EV}}_{\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathrm{EV}}_{\mathcal{A}} \text{ assert fails}]$$

Theorem 9.13 *Under the infrastructure assumptions, for honest setup authority, honest printer authority and honest control component i, for any polynomially bounded adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{EV}_{\mathcal{A}} \leq \boldsymbol{Adv}^{pIR}_{\mathcal{B},\mathsf{H}}$$

where $\mathcal{B}$ is an adversary against the pre-image resistance of hash function H.

Intuitively, the advantage results from guessing the confirmation authentication, breaking the pre-image resistance of hash function H. The proof for the theorem is in section 9.4.3.

9.3.4.2 Eligibility Uniqueness

Eligibility uniqueness ensures every eligible voter can only confirm a single vote.

Eligibility uniqueness is guaranteed by the confirmation oracle of the honest control component: Before some Id is added to $\boldsymbol{\mathcal{C}}_{Conf}$, it is ensured that $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf}$, hence there cannot be two votes confirmed under the same Id. This is already reflected in the definition of Extract.

Theorem 9.14 *For honest control component i, for any adversary $\mathcal{A}$, it holds that*

$$\boldsymbol{Adv}^{EU}_{\mathcal{A}} = 0$$

Proof The honest control component will check whether for the presented Id a vote was already confirmed, and will deny any further confirmations.□

9.4 Game Hops

Given the instantiated game-based definitions, here we prove the theorems to hold which bound the respective attacker-advantage.

Our proof strategy modifies the games step by step, until the adversary cannot win the game any more. For each modification, we bound the advantage of the adversary. In the end, we collect the bounds; hence add all advantages of the adversary throughout the modifications.

9.4.1 Individual Verifiability

We observe that the attacker must get both checks of the voter to succeed for the game not to abort early. We modify the game correspondingly.

The first abort is skipped if and only if the modular addition of the partial vote verifications is as expected. We argue that the attacker either needs to submit exactly what was cast by the voter, or guess the partial vote verification of the honest control component. In technical terms, it holds that either ($Id = Id'$ and $VA = VA'$ and $C' = C'$) or the attacker guesses $vv^{(1)}_{C'}$.

We reflect this idea in the first modification of the game: When the attacker guesses $vv_{C'}^{(1)}$ (denoted as event $\mathbf{E}_1$), we abort the game.

Further, the second abort is skipped if and only if the modular addition of the partial confirm verifications is as expected. We argue that the attacker needs to submit exactly what the voter confirmed, or guess the partial confirm verification of the honest control component. In technical terms, it holds that either ($Id = Id''$ and $CA = CA'$) or the attacker guesses $cv^{(1)}$.

We implement the second idea with the second modification of the game: When the attacker guesses $cv^{(1)}$ (denoted as event $\mathbf{E}_2$), we abort the game.

Both modifications together result in Game 13.

$\mathbf{Exp}^{\text{IV-1}}_{\mathcal{A}}(params)$:
$(v, cc, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

// the state of the honest voter 1
$(Id, VA, \{VV_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, CA, CV, PtC_{Id}) \leftarrow v_1$

// the attacker receives the private state of the dishonest voters and control components and is able to pick the honest voter's plain vote
$\mathcal{P} \leftarrow \mathcal{A}(v_{\ominus 1}, cc_{\ominus 1})$

// the voter picks the corresponding $\mathcal{C}$ and casts the vote
$\mathcal{C} \xleftarrow{} (\mathcal{P}, .) \in PtC_{Id}$

// the attacker decides what to forward to the honest control component
$(VA', \mathcal{C}') \leftarrow \mathcal{A}(VA, \mathcal{C})$
if $(VA' \neq VA$ or $\mathcal{C}' \neq \mathcal{C})$ then $\mathbf{E}_1 \leftarrow$ `true` abort

// the honest control component stores the cast vote and answers
if $((Id, \mathsf{Hash}(VA')) \in \boldsymbol{hVA}$ and $\mathcal{C}' \in \mathbb{C}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Cast})$
then $\boldsymbol{\mathcal{C}}_{Cast} \leftarrow \boldsymbol{\mathcal{C}}_{Cast} \cup \{(Id, \mathcal{C}')\}$; $(..., \{vv^{(1)}_{\mathcal{C}} \mid \mathcal{C} \in \mathbb{C}\}, ...) \leftarrow b^{(1)}_{Id}$; $\mathcal{A}(vv^{(1)}_{\mathcal{C}'})$

// the attacker decides what to forward to the voter
$(\{vv'^{(1)}_{\mathcal{C}}, ..., vv'^{(m)}_{\mathcal{C}}\}) \leftarrow \mathcal{A}()$

// the voter checks the vote verification
if $(VV_{\mathcal{C}} \neq \oplus^{m}_{i=1} vv'^{(i)}_{\mathcal{C}})$ then abort

// the voter confirms the vote
$\mathcal{A}(CA)$

// the attacker decides what to forward to the honest control component
$(CA') \leftarrow \mathcal{A}()$
if $(CA' \neq CA)$ then $\mathbf{E}_2 \leftarrow$ `true` abort

// the honest control component stores the confirmed vote and answers
$\mathcal{C}'' \leftarrow (Id, .) \in \boldsymbol{\mathcal{C}}_{Cast}$
if $((Id, \mathsf{Hash}(CA')) \in \boldsymbol{hCA}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf})$
then $\boldsymbol{\mathcal{C}}_{Conf} \leftarrow \boldsymbol{\mathcal{C}}_{Conf} \cup \{(Id, \mathcal{C}'')\}$; $(..., cv^{(1)}, ...) \leftarrow b^{(1)}_{Id}$; $\mathcal{A}(cv^{(1)})$

// the attacker decides what to forward to the voter
$(\{cv'^{(1)}, ..., cv'^{(m)}\}) \leftarrow \mathcal{A}()$

// the voter checks the vote verification
if $(CV \neq \oplus^{m}_{i=1} cv'^{(i)})$ then abort

// the attacker wins if $\mathcal{P}' \neq \mathcal{P}$
$\mathcal{C}''' \xleftarrow{} (Id, .) \in \boldsymbol{\mathcal{C}}_{Conf}$
$\mathcal{P}' \xleftarrow{} (., \mathcal{C}''') \in PtC_{Id}$
assert $\mathcal{P}' = \mathcal{P}$

Game 13: Modified game for individual verifiability. We let the adversary choose $\mathcal{P}$, the plain vote of the voter. Note that $\mathcal{C}'''$ as well as $\mathcal{P}'$ might be undefined.

Lemma 9.15 *Given the same security assumptions in both games, for any adversary $\mathcal{A}$, it holds that*

$$|\boldsymbol{Adv}_{\mathcal{A}}^{IV} - \boldsymbol{Adv}_{\mathcal{A}}^{IV\text{-}1}| \leq \frac{1}{|\mathbb{Z}_v|} + \frac{1}{|\mathbb{Z}_v|}$$

Proof As $\mathbf{Exp}_{\mathcal{A}}^{\text{IV}}$ and $\mathbf{Exp}_{\mathcal{A}}^{\text{IV-1}}$ are equal, except if either $\mathbf{E}_1$ or $\mathbf{E}_2$ occurs, we get that[18]:

$$|\mathbf{Adv}_{\mathcal{A}}^{\text{IV}} - \mathbf{Adv}_{\mathcal{A}}^{\text{IV-1}}| \leq \mathbf{E}_1 + \mathbf{E}_2$$

When the voting phase starts, the adversary does not know the secret state of the honest voter and the honest control component (see lemma 9.1 and lemma 9.3). Even when the adversary is passed the secret state of the other voters[19], this results in no advantage to guessing $b_{Id}^{(i)}$ and BS_{Id} directly (see lemma 9.2).

If $\mathbf{E}_1$ occurs, but the adversary still wins the game, then it must guess $vv_{\mathcal{C}'}^{(1)}$ such that $VV_{\mathcal{C}} = \oplus_{i=1}^{m} vv_{\mathcal{C}}^{\prime(i)}$, else the voter would abort the game. The value is picked uniform at random out of $\mathbb{Z}_v$ (see GenBallots), hence the attacker has no advantage to random guessing. Therefore, $\mathbf{E}_1 = \frac{1}{|\mathbb{Z}_v|}$.

If $\mathbf{E}_2$ occurs, but the adversary still wins the game, then it must guess $cv^{(1)}$ such that $CV = \oplus_{i=1}^{m} cv^{(i)}$, else the voter would abort the game. The value is picked uniform at random out of $\mathbb{Z}_v$ (see GenBallots), hence the attacker has no advantage to random guessing. Therefore, $\mathbf{E}_2 = \frac{1}{|\mathbb{Z}_v|}$. □

We now note that $\mathbf{Adv}_{\mathcal{A}}^{\text{IV-1}} = 0$. Indeed, it now holds that $VA = VA'$ and $\mathcal{C} = \mathcal{C}'$, hence the correct vote is cast (else, $\mathbf{E}_1$). Further, it now holds that $CA = CA'$, hence the correctly cast vote is also confirmed (else, $\mathbf{E}_2$). It must therefore hold that $\mathcal{C}''' = \mathcal{C}$, which also implies that $\mathcal{P} = \mathcal{P}'$.

Proof (theorem 9.6) Collecting the security assumptions and the bounds proofs theorem 9.6. □

9.4.2 Participation Verifiability - Participating Voter

$\mathbf{Exp}_{\mathcal{A}}^{\text{PVP}}$ is defined only relative to $\mathbf{Exp}_{\mathcal{A}}^{\text{IV}}$. Similarly, we will describe the participation verifiability proof only relative to the individual verifiability proof (see section 9.4.1).

[18] By Shoop's difference lemma.

[19] We omit [... and of the dishonest control components] as this was already known to the adversary from the setup phase.

Compared to $\mathbf{Exp}^{\text{IV}}_{\mathcal{A}}$, the only difference in $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$ is in the assertion: We no longer require the plain vote to match the voter's selection, but only require *some* confirmed vote to be reflected on the bulletin board. This make the attack harder for the adversary.[20]

We argue that we only need to apply the second modification of $\mathbf{Exp}^{\text{IV}}_{\mathcal{A}}$ to our $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$: When the attacker guesses $cv^{(1)}$ (denoted as event $\mathbf{E}_2$), we abort the game. This modification results in $\mathbf{Exp}^{\text{PVP-1}}_{\mathcal{A}}$.

Lemma 9.16 *Given the same security assumptions in both games, for any adversary $\mathcal{A}$, it holds that*

$$|\boldsymbol{Adv}^{PVP}_{\mathcal{A}} - \boldsymbol{Adv}^{PVP\text{-}1}_{\mathcal{A}}| \leq \frac{1}{|\mathbb{Z}_v|}$$

Proof As $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$ and $\mathbf{Exp}^{\text{PVP-1}}_{\mathcal{A}}$ are equal, except if $\mathbf{E}_2$ occurs, we get that:

$$|\mathbf{Adv}^{\text{PVP}}_{\mathcal{A}} - \mathbf{Adv}^{\text{PVP-1}}_{\mathcal{A}}| \leq \mathbf{E}_2$$

As argued before in section 9.4.2, it holds that $\mathbf{E}_2 = \frac{1}{|\mathbb{Z}_v|}$. □

We now note that $\mathbf{Adv}^{\text{PVP-1}}_{\mathcal{A}} = 0$. Indeed, it holds that $CA = CA'$, hence the voter has definitely confirmed their vote (else, $\mathbf{E}_2$).[21]

Proof (theorem 9.7) Collecting the security assumptions and the bounds proves theorem 9.7. □

9.4.3 Eligibility Verifiability

We observe that the attacker only succeeds if it manages to guess CA' such that $\mathsf{Hash}(CA') = \mathsf{Hash}(CA)$. We modify our game correspondingly.

We modify the game as follows: Whenever the attacker outputs CA' such that $\mathsf{Hash}(CA') = \mathsf{Hash}(CA)$, we denote this as the event $\mathbf{E}_3$ and abort the game.

The modification results in Game 14.

Lemma 9.17 *Given the same security assumptions in both games, for any polynomially bound adversary $\mathcal{A}$, it holds that*

$$|\boldsymbol{Adv}^{EV}_{\mathcal{A}} - \boldsymbol{Adv}^{EV\text{-}1}_{\mathcal{A}}| \leq \boldsymbol{Adv}^{pIR}_{\mathcal{B},\mathsf{H}}$$

[20] For example, if the attacker manages to successfully confirm $\mathcal{P}' \neq \mathcal{P}$ then the attacker would always win $\mathbf{Exp}^{\text{IV}}_{\mathcal{A}}$, but not necessarily in $\mathbf{Exp}^{\text{PVP}}_{\mathcal{A}}$.

[21] The vote itself might be undefined if the adversary prevented it being cast, but this does not make a difference for the definition here.

$\mathbf{Exp}^{\text{EV-1}}_{\mathcal{A}}(params)$:
$(\boldsymbol{v}, \boldsymbol{cc}, bb) \xleftarrow{r} \mathsf{Setup}_{\mathcal{A}}(params)$

// the id of the honest voter 1
$(Id, ..., CA, ...) \leftarrow v_1$

// the attacker receives the private state of the dishonest voters and control components
$\mathcal{A}(\boldsymbol{v}_{\ominus 1}, \boldsymbol{cc}_{\ominus 1})$

// the attacker decides how to query the honest control component
$(CA') \leftarrow \mathcal{A}()$
if $\mathsf{Hash}(CA') = \mathsf{Hash}(CA)$ then $\mathbf{E}_3 \leftarrow$ `true` abort

// the honest control component stores the confirmed vote
$\mathcal{C} \leftarrow (Id, .) \in \boldsymbol{\mathcal{C}}_{Cast}$
if $((Id, \mathsf{Hash}(CA')) \in \boldsymbol{hCA}$ and $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf})$
then $\boldsymbol{\mathcal{C}}_{Conf} \leftarrow \boldsymbol{\mathcal{C}}_{Conf} \cup \{(Id, \mathcal{C})\}$

// the attacker wins if some vote is confirmed
assert $(Id, .) \notin \boldsymbol{\mathcal{C}}_{Conf}$

Game 14: Game for eligibility verifiability. We let the adversary choose $\mathcal{P}$, the plain vote of the voter.

Proof As $\mathbf{Exp}^{\text{EV}}_{\mathcal{A}}$ and $\mathbf{Exp}^{\text{EV-1}}_{\mathcal{A}}$ are equal, except if $\mathbf{E}_3$ occurs, we get that:

$$|\mathbf{Adv}^{\text{EV}}_{\mathcal{A}} - \mathbf{Adv}^{\text{EV-1}}_{\mathcal{A}}| \leq \mathbf{E}_3$$

We observe that the adversary does not know CA when the voting phase starts (same argumentation as in proof 11). We further observe that CA appears uniform at random to the adversary (see lemma 9.1).

$\mathbf{E}_3$ occurs when the adversary guesses CA' such that $\mathsf{Hash}(CA') = \mathsf{Hash}(CA)$. The adversary is given $hCA = \mathsf{Hash}(CA)$, which corresponds to the pre-image resistance attacker $\mathcal{B}$ against Hash which advantage we have bound to $\mathbf{Adv}^{\text{pIR}}_{\mathcal{B},\mathsf{H}}$. Hence, $\mathbf{E}_3 \leq \mathbf{Adv}^{\text{pIR}}_{\mathcal{B},\mathsf{H}}$. □

We now observe that $\mathbf{Adv}^{\text{EV-1}}_{\mathcal{A}} = 0$. Indeed, when $\mathsf{Hash}(CA') = \mathsf{Hash}(CA)$ the game aborts. The adversary is therefore unable to confirm any vote for the voter under attack.

Proof (theorem 9.13) Collecting the security assumptions and the bounds proofs theorem 9.13. □

9.5 Summary

We first described the security guarantees we expect from our infrastructure, and how they impact our proofs. We assume perfect secure channels in the setup phase, and perfect authentic channels throughout all phases. We need a pre-image resistant hash function. The bulletin board stores the append-only state, communicated over authentic channels. The verifiable shuffle provides secure key generation and encryption, as well as a secure and correct decryption-shuffle. Besides the verifiable shuffle, no infrastructure assumption needs to be represented in the proofs.

Then we implemented the syntax as required by our formal model. We map the setup phase into the setup function, and prove it to provide secure initial state to the honest voter and honest control component. We implement the protocol state queries using the state of the honest control component. We further map the tally phase into the tally function, and prove the tally to be correct and private.

Finally, we proved property by property. Individual verifiability is guaranteed as long as the verification codes are suitably long. Participation verifiability additionally requires the pre-image resistant hash function (and therefore suitably long authentication codes). Vote verifiability is given by design. Universal verifiability requires the correctness of the shuffle. Vote secrecy is guaranteed by the secure encryption of the verifiable shuffle. Fairness is implied by vote secrecy. Eligibility verifiability holds due to the pre-image resistance of the hash function. Finally, eligibility uniqueness is again given by design.

Chapter 10

Conclusion

In this thesis, we have proposed a system for vote électronique. Its unique characteristic, compared to previous proposals of such schemes in Switzerland, is the use of code voting: The voter enters a code corresponding to their voting choice instead of the plain voting choice.

This small change not only increases security as the voter no longer needs to trust their own device, it also leads to a drastically simpler system concerning specification and proofs than previous endeavours. It is to be expected that also its implementation and operation will be less complex and error-prone.

We proved the system secure using a formal model adapted specifically to Switzerland, which translates the requirements set forward by law into formal game-based definitions. We identified properties not recognised in the literature, and improved other properties such that they apply to a wider range of electronic voting protocols. We use computational proofs to show that our proposal indeed satisfies the formal properties.

To ensure the system does not only meet current legislative requirements, but will likely also support future to-be-expected changes in law, we base the design of the scheme on an informal argument for optimality. We performed a survey of the literature and of history, law and politics in Switzerland concerning vote électronique. We then reflected upon the legislation, discussing how it may be adapted to lead to securer systems. Based on this new strongest currently achievable setting, we iteratively constructed a voting protocol which forms the basis of our new system.

As part of this work, we have also identified room for improvement within the legislative basis and within the Swiss Post scheme, which we have communicated with the responsibles.

Future work The voting protocol is fully specified, as is a first draft of the computational proofs. While the result at the very least convince of the *feasibility* of such a code voting system for vote électronique, we did have to restrict the scope of the thesis to fit in the six months available.

There is still some considerable work left until the proposed system might see real-world usage.

- Thorough review of the protocol and the computational proofs by independent researchers.
- Symbolic proof of the properties, notably as required by law.
- Specification of the infrastructure, crucially the verifiable shuffle and the bulletin board.
- Discussion about parameter sizes, based on the adversary bounds established in the computational proofs.
- Study about usability, exploring how the code voting mechanism as proposed performs best in practice.

Besides future work specific to the proposed system, we have also identified some gaps in the literature which motivate future research.

- Survey about mechanisms known in the literature (like section 3.3).
- Discussion about achievable properties under specific circumstances (like chapter 5).
- Optimality arguments of constructions (like chapter 6).
- Formalization of properties which match more complex voting protocols (like chapter 8).

Appendix A

Trivia

For the Norways 2011 election, when printing the voting cards, it was noticed late that unfortunately some parties were missing. The printing was stopped, and the protocol setup had to be redone. However, access to the server was denied: The terrorist attack of Andres Breivik impacted the building which housed one of the servers, and the crime scene was under lockdown. After a few days, they finally got permission to evacuate the server, and could restart the printing. Additionally endangering the deadline, the printing company noticed the printing took longer than expected, but the voting period could not start until printing was finished. Three shifts worked around the clock to compensate, and the system went online with only a few hours delay [128].

For the Norways 2013 elections, the Internet Election Committee (IEC) was founded, composed out of voters, experts and representatives from the administration. A counting ceremony was held to make the tallying of votes somewhat approachable. The result of the ceremony was discarded [186].

For the Norways 2013 elections, the IEC announced they would like to accommodate media or political representatives to conduct verification. No party showed interest, so the only organization which performed verification was contracted by the government itself [213].

Voting booths in France have an error rate (difference between amount of signatures and counted votes, which should be 0) that is 4 to 6 times higher with electronic counting machines than with paper ballots. No convincing hypothesis explains this gap [107].

Sweden wanted to start with remote voting trials for their election in 2019. However, their trial was stopped a week before the vote, presumably due to reviews arriving late and the responsible political side having lost confidence [104]. Scytl implies it was stopped to due GDPR concerns [250].

In the Netherlands, following a suddenly heated public debate, the minister forbid usage of all software one month before the election of 2017. Most importantly impacted was the supporting software for the count, which had organically grown over the years to take over more and more tasks. Suddenly, the municipalities needed to reorganize how they would perform the election, but their questions to the ministry remained unanswered (Could they still use computers? Could they use spreadsheets?). Under pressure of the municipalities, and after a report found only minor issues with the supporting software, the ministry again allowing usage of the software, under the condition that all counts had also to be performed manually. Everything then seemed to go smoothly, however after the election some discrepancies between municipality results and aggregated results were observed: A few thousand votes went missing. An investigation revealed that a system crash lead to data loss, and due to time constraints, the clerks did not double-check their inputs [65].

Appendix B

Swiss Post Protocol

For a summary of this protocol see section 3.2.3. Each paragraph in the summary corresponds to one section here, and phrase in the summary corresponds to one paragraph here.

Publicly known for each election e are the N voters participating, the n available voting options and the amount ψ of voting options selectable. The valid combination of voting options is defined using a list of ψ entries, with each being a $correctnessID$ which differs for different questions.[1] The voter communicates over the voting device to the voting server. The voting server handles communication between all other server-side entities. All server-side entities append cryptographic values to the log, which is used by the auditors to verify correct execution. To setup the protocol, a fully trusted printer prepares the election key material.[2]

B.1 Setup

Each return code control component (CCR) chooses a secret and ψ encryption key pairs. The public keys are added to the log and then sent to the printer. The printer aggregates them into the choice return code public keys (resulting in a single key per selectable voting option). The printer logs the received and aggregated public keys.

The printer chooses n setup key pairs, and logs their public keys. For each voter, the printer chooses an id vcd, a verification card key pair (sk, pk) and a ballot casting key bCK.

[1] For example, for an election with a yes/no question and a choose-three question, the table would look like (a, b, b, b).

[2] The terminology "printer" is likely only used to comply with the Swiss regulation, which in its version until 2021 does not know a setup component (but a fully-trusted printer).

Each of the n voting options (encoded into primes) is raised to sk to result in pCC. bCK is squared and raised to sk to result in the confirmation key CK. Each pCC and the single CK is hashed and squared to result in many $hpCC$ and a single hCK. $hpCCHash$ are generated by hashing each $hpCC$ with vcd, e and its corresponding entry in the $correctnessID$ table.

Each $hpCC$ is encrypted by one if the n setup keys, and hCK is encrypted by the aggregated setup keys. vcd, pk, the encrypted $hpCC$s and hCK, and a shuffled list of the $hpCCHash$ are sent to each CCR and appended to the logs.

Each CCR derives two voter secrets using a KDF, the first by using its secret and vcd as a seed, the second by additionally appending the constant "confirm" to the previously used seed. Each encrypted $hpCC$ is raised to the first voter secret, the encrypted hCK is raised to the second. The resulting values, a zero-knowledge proof of the exponentiations, and the generator raised to each of the secrets[3] is returned to the printer.

The printer now multiplies the resulting values of each CCR, then decrypts them using the setup secret keys into the many pC and the single $pVCC$. These are then hashed into $lCC = H(pC, vcd, e, correctnessID)$ and $lVCC = H(pVCC, vcd, e)$.

The printer now picks randomly for each voter n short choice return codes CC and a single short vote cast return code VCC. They are associated to the long equivalents by a key-value store: The key is given by the hash of the long code, the value is an encryption of the short code, under a key derived out of the long code.[4] The key-value store is shuffled, appended to the log, and sent to the voting server.

Finally, the printer chooses a short voter-specific secret, uses it as a seed to a KDF to derive a symmetric key, and then encrypts the verification card secret key into a key store. The store is appended to the log, and sent to the voting server. The short secret is sent to the voter, who can use it to download and unlock the store.[5]

Each mixing control component (CCM) - except the last CCM which remains offline - chooses an election key pair, appends the public key to the log and sends it to the printer. The printer chooses the electoral board key pair. Shares of the private key are distributed to trustees. The public key is aggregated with the public keys of the CCMs into the the election public key. All public keys are appended to the log, and the election public key is sent to the voting server.

[3] Essentially resulting in two public keys, each derived out of one of the secrets.

[4] Note that this construction allows to retrieve the short code iff the long code is known.

[5] This improves usability for the voter, as the secret can be chosen much shorter than the full verification card secret key.

The setup phase finishes with the auditors checking the zero-knowledge proofs in the log.

B.2 Voting

The voter receives a voting card with its short secret (for the key store), the short choice return codes CC for each choice, the ballot casting key bCK and the short vote cast return code VCC. The voter authenticates[6], downloads the key store, and unlocks the verification card secret key sk.

The voter multiplies the selected voting options and encrypts them under the election public key into $E1$. Each voting option is further raised to sk, and encrypted under the choice return code public keys into the list $E2$. $E1$ and $E2$ are sent to the voting server, together with zero-knowledge proofs asserting that $E1$ and $E2$ contain the same encrypted voting options.

The voting server forwards the values to the CCRs. Each CCR ensures the voter has not voted yet, checks that the vote contains the expected number of selections and verifies the proofs. If all succeeds, the vote is registered as cast. Each CCR now uses its set of encryption key pairs to partially decrypt $E2$[7], and adds the result together with zero-knowledge proofs of correct exponentiation to the log.

Now each CCR verifies the proof of the others, then combines the partial decryptions to fully decrypt $E2$ into the pCC's[8]. The pCC's are added to the log. Each CCR generates from each pCC the respective $hpCC$ and $hpCCHash$ as done in the setup phase. When all $hpCCHash$ exist as expected, each $hpCC$ is raised to the appropriate voter secret to recover the partial lCC. Both pCC and the partial lCC, as well as a proof of correct exponentiation, are added to the log.

The voting server now multiplies each the partial lCC into pC to then derives each full lCC as done in the setup phase. Finally, the short choice return code CC belonging to each lCC is extracted from the key-value store. All are added to the log, and then returned to the voter.

The voter compares the received CC with the one printed on the voting card, and if they match, the voter enters bCK. The voting client computes the CK and sends it to the voting server.

Each CCR ensures the confirmation procedure is for a valid vote, and has not been attempted to many times. It hashes and squares CK to compute hCK,

[6] Authentication is not specified yet as of Version 0.9.10.

[7] As multi-recipient ElGamal is used as the encryption here, this means raising the first term of the encryption to the secret.

[8] As multi-recipient ElGamal is used as the encryption here, this means dividing the second term by a multiplication of all first terms.

return codes	finalization code
$pCC = p^{sk}$	$CK = bCK^{2*sk}$
$hpCC = H(pCC)^2$	$hCK = H(CK)^2$
$lCC_i = hpCC^{vs_{i,1}}$	$lVCC_i = hCK^{vs_{i,2}}$
$pC = \prod lCC_i$	$pVCC = \prod lVCC_i$
$lCC = H(pC, vcd, ee, correctnessID)$	$lVCC = H(pVCC, vcd, ee)$
$CC = Dec(lCC, st[H(lCC)])$	$VCC = Dec(lVCC, st[H(lVCC)])$

Table B.1: Overview of the operations performed on the return codes and the finalization code over all components. For $vs_{i,1}$ and $vs_{i,2}$ the voter secrets of the CCR i, and st the respective key-value store for the encrypted short codes. The variable names are the same as used in the specification, although casing has been changed to be little bit closer to consistent naming. Note the naming collision with pC of line 4, which should have been named pCC (compare with the right column), but this was already defined on line 1.

and then raises the result to the second voter secret to get the partial $lVCC$. $lVCC$ is appended to the log, together with CK, the number of attempts, and a zero-knowledge proof of the exponentiation.

The voting server now multiplies all partial $lVCC$ into $pVCC$ (same as in the setup phase) to ultimately derive the full $lVCC$. Finally, the short vote cast return code VCC belonging to the $lVCC$ is extracted from the key-value store, added to the log together with the attempts count, and returned to the voter. The voter compares the VCC with the one printed on the voting card.

The voting phase finishes with the auditors checking consistency of the values in the log, specifically that the CCR logs are equal, the recorded votes are consistent, the zero-knowledge proofs verify and the extraction of the short codes are reproducible.

B.3 Tallying

The voting server prepares the tally by extracting the ciphertext $E1$ from all confirmed votes. Each CCM, one after the other, mixes the ciphertexts and partially decrypts. Both mix proofs as well as a zero-knowledge proof of partial decryption are stored in the log. The auditors verify the proofs, and if successful, the trustees reconstruct their electoral board private key. The key is passed to the last offline CCM, which does the final decryption after another mix. The now fully decrypted plain votes are factorized into their composing primes to recover the selected voting options.

The tally phase finishes with the auditors checking the execution of the last CCM: The mix proof and the proofs of decryption are verified, and the factorization of the plain votes is reconstructed [235].

Appendix C

Use auditors to check trust assumptions

The auditors are not strictly required for the protocol to reach its security properties under the trust assumptions (see chapter 7). One might however employ auditors to give additional assurances that the trust assumptions are not broken.

The description here is informal, and there needs to be extra care employed when designing appropriate procedures that the checks are effective (meaning the attacker cannot simply circumvent it) and do not break the security properties (for example by giving the auditors too much power like exposing inadequate secret material).

Setup phase Auditors can check that Id_1 actually does not lead to overlaps with other elections and $\boldsymbol{Id}$ is generated as expected. Further, auditors can check that *PtC* encodes each potential voting option into a separate code. Additionally, auditors may perform basic consistency checks, like $|\mathbb{V}| = |\boldsymbol{Id}| = |\boldsymbol{hVA}| = |\boldsymbol{hCA}|$, $\forall Id \in \boldsymbol{Id}.(Id, .) \in \boldsymbol{hCA}$ and $\forall Id \in \boldsymbol{Id}.(Id, .) \in \boldsymbol{hVA}$. Further, the auditors also check that all cryptographic parameters make sense, e.g. $\mathbb{Z}_a$ and $\mathbb{Z}_v$ to choose the authentication and verification secrets.

To audit the second part of the setup phase, auditors may be empowered to choose some voters $\mathbb{V}' \subset \mathbb{V}$ of which they audit their ballot sheets, to assert the setup and the printer component operate as expected. It is important that the chosen voters $\mathbb{V}'$ will not participate in the elections later on[1] (as the auditors learn their secrets), and that the selection is not predictable by the attacker (as then the check would be ineffective). Per our setting, there

[1] Of course, excluding real voters from the election is not a good option. One could instead empower the auditors to create fake voters, however these must not be distinguishable from real voters.

may be only a single honest auditor, hence each auditor must select fake voters by themselves and in secret.

When the ballot sheets finished printing - but of course before sending them out - each auditor reveals the voters it wants to audit.[2] The ballot sheets of all voters under audit are then examined by all auditors together.

The auditors may check the *Id* is valid and the authentication secrets are correctly represented as hashes on the bulletin board. Further, they may ensure each plain voting option is assigned a different code, and this code is indeed in $\mathbb{C}$.

The auditors can further also check that the verifiable shuffle preparation has been done correctly, as the randomness used when creating the shuffle input ciphertexts is also part of the ballot sheet. For each code vote, the auditors ensure the appropriate ciphertext represents the expected plain vote.

Going even further, the auditors may also verify that the partial ballot combination has been done correctly. For this, they request the partial ballot of the voters under audit directly from the control components.[3] Then, they check whether the combination of the partial ballots indeed result in the given ballot sheet.

The auditors may also do cross-checking of ballot sheets, for example to ensure the plain to code lookup is reasonably different on each ballot sheet.

Voting phase Auditors can continuously check the bulletin board for consistency. Concretely, they can ensure for each cast vote that the cast authentication is published. Further, for each confirmed vote, the auditors can check the confirm authentication is published, and the vote has been correctly cast.

Auditors may also verify that the bulletin board indeed processes new input, and the control component also react to the new messages. For this purpose, the auditors may be empowered to choose additional voters $\mathbb{V}'' \subset \mathbb{V}$, again choosing them in secret and individually, but then performing the actions together. When a vote is confirmed during the audit, of course the represented plain vote has to be subtracted from the final result again.

Tally phase During the tally phase, the verifiable shuffle executed by the control components produce proofs. The auditors may verify these proofs are indeed correct. Further, they may assert the final announced result indeed is the sum of the plain votes resulting from the shuffle.

[2] The process needs to ensure auditors indeed reveal all their chosen voters.

[3] Note that the corresponding request has to be placed on the bulletin board, to establish an audit trail. Else individual auditors might deceive control components to reveal secrets of voters not under audit.

Appendix D

Differences of computational proof to other computational proofs of vote électronique protocols

We explicitly want to document the differences between the computational proofs of vote électronique protocols.

D.1 CHVote Verifiability Analysis

The document only takes the verifiability properties of CHVote into account. It first establishes clear definitions and then presents both computational and symbolic proofs [24]. The document is clearly structured, and the proofs are comparatively easy to follow. It was written by established researchers in the field. Throughout this section, we refer to [24] as the *analysis*.

In general, the overall structure of the properties and corresponding game-based definitions are similar. We justify here notable differences in modelling, notation and definitions.

The analysis uses the single-pass voting model, however this model does not really fit the two-roundtrip scheme we intend to prove. We use a somewhat more abstract model which defines less about the messages exchanged of the voting protocol, but instead provides functions to directly assess the state of the voting protocol.

Overall, our game-based definitions are less specialized to the specific properties, but instead always preserve the same structure. This aims to simplify the model (as few functions have to be assumed) and the proofs (as the structure can be reused). Further, our games are more consistent, which is a valid design goal as arguably only all properties together result in a useful - as secure - voting system.

As a last general change, we adopt the notation to align closer to the naming used already in the proposal. Instead of election authorities, we have control components. Instead of *sk* and *vsk* for the private state of voters and control components, we refer to *v* and *cc*. Instead of *pk*, we refer to the public state as *bb*, the bulletin board state. Our naming overall aims to stronger distinct the different roles and their values.

Difference by property In the individual verifiability game, the analysis provides the attacker with both the public information as well as the private state of the corrupted control components $\boldsymbol{cc}_{\ominus 1}$ to decide upon the voter selection *s*. We argue this is inconsistent; if the attacker is provided with private state at this point of the game (concretely $\boldsymbol{cc}_{\ominus 1}$), then it should be provided with all private state it eventually gets access to (notably, including the private state of the corrupted voters $\boldsymbol{v}_{\ominus 1}$). We reflect this in our definitions.

Participation verifiability or something similar is not mentioned by the analysis. We note that this is also not prominently claimed by CHVote.

For vote verifiability[1] the attacker is allowed full access to the setup procedure. We however reuse the same game structure as for individual verifiability, explicitly including the setup procedure. We argue that this does not overall weaken our definition, and it is hard to imagine a voting protocol which can decide whether a vote is valid without any assumptions on the setup. Indeed, when proving the property for CHVote, the analysis again includes the setup procedure with an honest control component.

In the universal verifiability game, the analysis lets the adversary generate a valid bulletin board and claim a result. If the result cannot be disproven (but is still wrong), the attacker wins. We take over the latter part, but again replace the former part with the same structure as in the individual verifiability game. By forcing the attacker to use the normal setup procedure and play against the actual bulletin board, we avoid defining a possibly complex validation function over the bulletin board (as by design only valid bulletin boards are produced).

Voter secrecy and fairness are out of scope for the analysis.

For eligibility verifiability, the attacker in the analysis wins if the bulletin board contains a confirmed vote of some voter which does not reflect the voter's intention. This implies that an adversary may confirms a vote, as long as it reflects the intention of the corresponding voter. However, this is too weak, as voters must be aware whether they have participated in the

[1] Referred to as ballot verifiability in the analysis.

election or not.[2] We therefore require the stronger notion that if a voter has not confirmed their vote, then there must be no corresponding confirmed vote on the bulletin board.

Eligibility uniqueness is defined identically by the analysis.

D.2 Scytl / Swiss Post Computational Proof

The document includes both a formal description of the protocol as well as the computational proof for both verifiability and privacy properties [235]. It was initially written by Scytl, then extended by Swiss Post once they took over development. Throughout this section, we refer to [235] as the *analysis*.

The analysis defines the properties directly protocol-specific.

Difference by property The analysis does not proof individual verifiability, but two sub-properties instead: *Sent-as-intended* and *recorded-as-confirmed - vote rejection*. The first property ensures the vote verification codes cannot be produced by the adversary, the second property ensures the same for the confirm verification.[3] In both games, the adversary does not choose the voter selection, it is chosen at random instead.

Universal verifiability is directly proven. The analysis lets the adversary generate a valid bulletin board and claim a result. If the result passes validation, but is different to the real result, the attacker wins.

[redacted][4] Swiss Post has been made aware of this finding.[5]

A notion of vote verifiability is proven, called *vote compliance*, where the attacker tries to cast an invalid ballot.

Vote secrecy is proved based on an adapted BPRIV definition (see [23]).[6] The attacker is provided with additional oracles for each message exchanged be-

[2] In Switzerland, voters may use different voting channels, but double-voting is forbidden. Voters must therefore know whether an electronic vote was already cast in their name.

[3] In the analysis, these values are called choice return code and vote cast code.

[4] [redacted]

[5] https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/issues/31

[6] The ideal model as assumed by the BPRIV definition however does not match the ideal model the Swiss Post protocol achieves. For example, in the Swiss Post protocol, the attacker additionally learns whether voters attempt to confirm their vote, based on observing the number of messages exchanged.

tween the honest voter and the honest control component. The $\mathcal{O}$cast oracle is not instantiated, and the side-conditions of BPRIV (strong consistency, strong correctness, authentication [23, 172]) are not explicitly proven to hold. The attacker is restricted in the sense that it is not allowed to break the finalization code mechanism, hence if the vote has been confirmed successfully by the control components, the voter individual verifiability check must also succeed.

Fairness is neither mentioned nor proven. The protocol likely fulfils the property; the vote is encrypted on the voter device and only decrypted in the tally phase.

Eligibility verifiability is proven as part as individual verifiability (denoted as *recorded-as-confirmed - vote injection*). Again, the voter selection is chosen at random. The attacker is provided with the data necessary to submit a vote, but not to confirm it. The adversary wins if it manages to confirm the vote without learning the confirm authentication[7] from the voter.

Eligibility uniqueness is not mentioned by the analysis. The protocol likely fulfils the property; the control components explicitly check for double-casting and its effects.[8]

[7] Called *BCK* in the analysis.

[8] For example *VerifyBallotCCR* aborts if the voter already registered some valid vote.

Appendix E

Privacy Definitions

As the privacy definitions presented in section 8.2.3 are somewhat novel, we argue here more explicitly why they are appropriate. We also note that while for vote secrecy (also sometimes called ballot privacy) there are many formal definitions in the literature, we did not encounter a formal definition for fairness.

Comparison to BPRIV Our definitions use similar ideas as the BPRIV definition [23, 172, 85], but BPRIV is intended for protocols matching the single-pass voting model. When a protocol does not satisfy the single-pass voting model, then it will also not match the ideal world the BPRIV definition provides guarantees to.[1]

Our definition is more general as it provides oracles to the adversary for each message processed or produced by honest instances, and therefore also specifically considers messages not part of the single-pass voting model. Consider an artificial voting scheme where the control components broadcast for each received ballot its decryption. Vote privacy is trivially broken, but instantiating BPRIV might miss the attack as the broadcast is not part of the single-pass voting model.

Further, our definition considers compromised voters, providing the adversary with some part of honest secret key material. Consider an artificial voting scheme encrypting the vote with a public key, but all honest voters know the corresponding secret key. Vote privacy is trivially broken as soon as the adversary compromises a voter, but BPRIV restricts the adversary to only learn public keys.

[1] Therefore the strong guarantees the definition provides for single-pass voting models, namely that satisfying BPRIV and some side conditions ensures the voting protocol is indistinguishable to some ideal world, does not apply any more.

Thirdly, our definition does not constrain the adversary at all; instead of only having two oracles available ($\mathcal{O}$voteLR and $\mathcal{O}$cast) to cast and confirm the vote, the adversary can now arbitrarily interact with the honest instances. Consider an artificial voting scheme which publishes voter secret material if a voter confirms their vote before casting it. Vote privacy can be broken by an adversary who reorders messages, but BPRIV might not capture it.

All these issues arise when the BPRIV definition is used without adding auxiliary oracles, which for example exchange further messages or compromise voters. However, when additional oracles are added, it is unclear which guarantees BPRIV still achieves, or more concretely, whether the proof into the ideal world can be adapted to incorporate the additional oracles.

Our definitions We construct our definitions by taking over two crucial ideas: First, the attacker is able to specify the voter selection in the real and in the fake world. Second, the tally always returns the tally result of the real world, possibly while simulating parts of the tally. As with our verifiability definitions, we do not explicitly specify all available oracles.

We motivate the definitions by informally reasoning about how relevant aspects are covered.

Any vote secrecy attacks involving a strict subset of control components or compromised voters is covered trivially, as the attacker is passed their private state. Vote secrecy attacks (ab)using some interaction pattern with the honest instances are also covered, as the adversary is given access to appropriate oracles. Finally, vote secrecy leaks in the tally phase are covered as the adversary is able to participate during tally.

Fairness attacks which rely on individually decrypted votes trivially lead to a win of the fairness game. Fairness attacks which may rely on a certain aggregation of votes[2] are also covered: As the adversary can arbitrarily cast and confirm votes (except for the honest voter) the adversary is able to simulate any aggregation which gives it an advantage.

[2]When a certain set of votes allows to draw conclusions over what tally they represent, possibly without knowing what any individual vote represents.

Bibliography

[1] Richard Adams. IT external audit: Final Report for the iVote system as implemented by the Western Australian Electoral Commission for the March 2017 State Election. report, Richard Adams, 2017.

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

[3] Ben Adida, Olivier De Marneffe, Olivier Pereira, Jean-Jacques Quisquater, et al. Electing a university president using open-audit voting: Analysis of real-world use of Helios. *EVT/WOTE*, 9(10), 2009.

[4] Ben Adida and C Andrew Neff. Ballot Casting Assurance. *EVT*, 6:7, 2006.

[5] Republik Adrienne Fichter. Die Post kann beim E-Voting weiterbasteln. https://www.republik.ch/2021/08/05/die-post-kann-beim-e-voting-weiterbasteln, November 2021. Accessed at 2022-01-20.

[6] Jordi Puiggalí Allepuz and Sandra Guasch Castelló. Internet voting system with cast as intended verification. In *International Conference on E-Voting and Identity*, pages 36–52. Springer, 2011.

[7] Jordi Puiggalí Allepuz and Sandra Guasch Castelló. Cast-as-intended verification in Norway. In *Proc. 5th Conf. Electron. Voting*, pages 49–63. Citeseer, 2012.

[8] Mohammed Alsadi and Steve Schneider. Verify My Vote: Voter Experience. *E-Vote-ID 2020*, page 280, 2020.

[9] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In *International Conference on Principles of Security and Trust*, pages 21–40. Springer, 2013.

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

[11] UN General Assembly et al. Universal declaration of human rights. *UN General Assembly*, 302(2):14–25, 1948.

[12] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography Conference*, pages 137–156. Springer, 2007.

[13] Jordi Barrat, Michel Chevalier, Ben Goldsmith, David Jandura, John Turner, and Rakesh Sharma. Internet voting and individual verifiability: the Norwegian return codes. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[14] David Basin, Saša Radomirovic, and Lara Schmid. Alethea: A provably secure random sample voting protocol. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 283–297. IEEE, 2018.

[15] David Basin, Saša Radomirović, and Lara Schmid. Dispute resolution in voting. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 1–16. IEEE, 2020.

[16] Christoph Baumgartner and Michael von Arx. Auditbericht: Sicherheitsüberprüfung betriebliche Aspekte E-Voting-System der Post. report, Baumgartner, Christoph and von Arx, Michael, July 2019. Accessed at 2021-05-19.

[17] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 263–280. Springer, 2012.

[18] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, 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.

[19] Josh Benaloh. Simple Verifiable Elections. *EVT*, 6:5–5, 2006.

[20] Josh Benaloh. Rethinking voter coercion: The realities imposed by technology. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[21] Josh Benaloh, Peter YA Ryan, and Vanessa Teague. Verifiable postal voting. In *Cambridge International Workshop on Security Protocols*, pages 54–65. Springer, 2013.

[22] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553, 1994.

[23] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516. IEEE, 2015.

[24] David Bernhard, Véronique Cortier, Pierrick Gaudry, Mathieu Turuani, and Bogdan Warinschi. Verifiability analysis of CHVote. report, Bernhard, David and Cortier, Véronique and Gaudry, Pierrick and Turuani, Mathieu and Warinschi, Bogdan, 2018.

[25] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 626–643. Springer, 2012.

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

[27] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *International Conference on E-Voting and Identity*, pages 111–124. Springer, 2007.

[28] Jurlind Budurushi, Stephan Neumann, Maina M Olembo, and Melanie Volkamer. Pretty understandable democracy-a secure and understandable internet voting scheme. In *2013 International Conference on Availability, Reliability and Security*, pages 198–207. IEEE, 2013.

[29] Jurlind Budurushi and Melanie Volkamer. Feasibility analysis of various electronic voting systems for complex elections. In *Conference for E-Democracy and Open Governement*, page 141, 2014.

[30] Christian Bull, Kristian Gjøsteen, and Henrik Nore. Faults in Norwegian internet voting. *E-Vote-ID 2018*, page 166, 2018.

[31] Christian Bull, Kristian Gjøsteen, Katrine Stemland Skjelsvik, Ida Sofie, and Gebhardt Stenerud. The imperfect storm. In *The International Conference on Electronic Voting: E-Vote-ID*, pages 116–120, 2016.

[32] Schweizerische Bundeskanzlei. Der Vote électronique in der Pilotphase: Zwischenbericht. report, Schweizerische Bundeskanzlei, August 2004. Accessed at 2021-05-18.

[33] Schweizerische Bundeskanzlei. Bericht über die Ergebnisse der Anhörung zum Erlass eines Technischen Reglements Vote électronique. https://www.bk.admin.ch/dam/bk-intra/de/dokumente/pore/politische_rechte/auswertender_berichtanhoerungveles.pdf.download.pdf/auswertender_berichtanhoerungveles.pdf, November 2013. Accessed at 2021-05-17.

[34] Schweizerische Bundeskanzlei. Bericht über die Ergebnisse der Anhörung zur Revision der Verordnung über die politischen Rechte (Vote électronique). https://www.bk.admin.ch/dam/bk-intra/de/dokumente/pore/politische_rechte/auswertender_berichtanhoerungvpr.pdf.download.pdf/auswertender_berichtanhoerungvpr.pdf, September 2013. Accessed at 2021-05-17.

[35] Schweizerische Bundeskanzlei. Verordnung über die elektronische Stimmabgabe (VEleS). https://www.fedlex.admin.ch/eli/cc/2013/859/de, December 2013. Accessed at 2021-05-17.

[36] Schweizerische Bundeskanzlei. Vote électronique: Menschen mit einer Behinderung können im Kanton Basel-Stadt erstmals elektronisch abstimmen . https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-60955.html, March 2016. Accessed at 2021-11-25.

[37] Schweizerische Bundeskanzlei. Bundesrat beschliesst nächste Schritte zur Ausbreitung der elektronischen Stimmabgabe. https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-66273.html, April 2017. Accessed at 2021-05-19.

[38] Schweizerische Bundeskanzlei. Schlussbericht: Expertengruppe elektronische Stimmabgabe EXVE. report, Schweizerische Bundeskanzlei, April 2018. Accessed at 2021-05-19.

[39] Schweizerische Bundeskanzlei. Vorentwurf Bundesgesetz über die politischen Rechte. https://fedlex.data.admin.ch/filestore/

fedlex.data.admin.ch/eli/dl/proj/6018/92/cons_1/doc_1/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-6018-92-cons_1-doc_1-de-pdf-a.pdf, December 2018. Accessed at 2021-05-17.

[40] Schweizerische Bundeskanzlei. Änderung des Bundesgesetzes über die politischen Rechte (Überführung der elektronischen Stimmabgabe in den ordentlichen Betrieb). https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/dl/proj/6018/92/cons_1/doc_2/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-6018-92-cons_1-doc_2-de-pdf-a.pdf, December 2018. Accessed at 2021-05-17.

[41] Schweizerische Bundeskanzlei. Bundeskanzlei nimmt Standortbestimmung zum E-Voting vor. https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-74508.html, March 2019. Accessed at 2021-05-19.

[42] Schweizerische Bundeskanzlei. E-Voting: Bundesrat richtet Versuchsbetrieb neu aus und stellt Einführung als ordentlicher Stimmkanal zurück. https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-75615.html, June 2019. Accessed at 2021-10-11.

[43] Schweizerische Bundeskanzlei. E-Voting: Bundesrat richtet Versuchsbetrieb neu aus und stellt Einführung als ordentlicher Stimmkanal zurück. https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-75615.html, June 2019. Accessed at 2021-05-19.

[44] Schweizerische Bundeskanzlei. Medienmittelung: Bundeskanzlei nimmt Standortbestimmung zum E-Voting vor. https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-74508.html, March 2019. Accessed at 2021-05-19.

[45] Schweizerische Bundeskanzlei. Vote électronique -Öffentlicher Intrusionstest 2019. report, Schweizerische Bundeskanzlei, August 2019. Accessed at 2021-05-19.

[46] Schweizerische Bundeskanzlei. Änderung des Bundesgesetzes über die politischen Rechte (Überführung der elektronischen Stimmabgabe in den ordentlichen Betrieb): Ergebnisbericht der Vernehmlassung. https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/dl/proj/6018/92/cons_1/doc_7/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-6018-92-cons_1-doc_7-de-pdf-a.pdf, June 2019. Accessed at 2021-05-17.

[47] Schweizerische Bundeskanzlei. Neuausrichtung und Wiederaufnahme der Versuche. report, Schweizerische Bundeskanzlei, November 2020. Accessed at 2021-05-21.

[48] Schweizerische Bundeskanzlei. Bund startet Überprüfung des neuen E-Voting-Systems . https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-84337.html, July 2021. Accessed at 2021-10-22.

[49] Schweizerische Bundeskanzlei. Teilrevision der Verordnung über die politischen Rechte und Totalrevision der Verordnung der BK über die elektronische Stimmabgabe (Neuausrichtung des Versuchsbetriebs): Ergebnisbericht der Vernehmlassung. https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/dl/proj/2021/61/cons_1/doc_9/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-2021-61-cons_1-doc_9-de-pdf-a.pdf, December 2021. Accessed at 2021-12-12.

[50] Schweizerische Bundeskanzlei. Teilrevision der Verordnung über die politischen Rechte und Totalrevision der Verordnung der BK über die elektronische Stimmabgabe (Neuausrichtung des Versuchsbetriebs): Erläuternder Bericht zur Vernehmlassung. report, Schweizerische Bundeskanzlei, April 2021. Accessed at 2021-05-17.

[51] Schweizerische Bundeskanzlei. Teilrevision der Verordnung über die politischen Rechte und Totalrevision der Verordnung der BK über die elektronische Stimmabgabe (Neuausrichtung des Versuchsbetriebs): Stellungnahmen. report, Schweizerische Bundeskanzlei, August 2021. Accessed at 2021-10-22.

[52] Schweizerische Bundeskanzlei. Vorentwurf Verordnung der BK über die elektronische Stimmabgabe(VEleS). https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/dl/proj/2021/61/cons_1/doc_2/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-2021-61-cons_1-doc_2-de-pdf-a.pdf, April 2021. Accessed at 2021-05-17.

[53] Schweizerischer Bundesrat. Verordnung über die politischen Rechte (VPR). https://www.fedlex.admin.ch/eli/cc/1978/712_712_712/de, May 1978. Accessed at 2021-05-17.

[54] Schweizerischer Bundesrat. Bericht über den Vote électronique: Chancen, Risiken und Machbarkeit elektronischer Ausübung politischer Rechte. report, Schweizerischer Bundesrat, February 2002. Accessed at 2021-05-18.

[55] Schweizerischer Bundesrat. Bericht über die Pilotprojekte zum Vote électronique. report, Schweizerischer Bundesrat, May 2006. Accessed at 2021-05-18.

[56] Schweizerischer Bundesrat. Bericht des Bundesrates zu Vote électronique. report, Schweizerischer Bundesrat, June 2013. Accessed at 2021-05-18.

[57] Schweizerischer Bundesrat. Ergänzende Dokumentation zum dritten Bericht des Bundesrates zu Vote électronique. report, Schweizerischer Bundesrat, June 2013. Accessed at 2021-08-06.

[58] Schweizerischer Bundesrat. Nationalratswahlen mit dem elektronischen Stimmkanal. https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-58314.html, August 2015. Accessed at 2021-05-19.

[59] Schweizerischer Bundesrat. E-Voting: Bundesrat lanciert Neuausrichtung des Versuchsbetriebs. https://www.bk.admin.ch/bk/de/home/dokumentation/medienmitteilungen.msg-id-81772.html, December 2020. Accessed at 2021-05-21.

[60] Schweizerischer Bundesrat. Neue rechtliche Grundlagen für Versuche mit E-Voting sollen Mitte 2022 vorliegen . https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-86387.html, December 2021. Accessed at 2021-12-12.

[61] Schweizerischer Bundesrat. Vorentwurf Verordnung über die politischen Rechte(VPR). https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/dl/proj/2021/61/cons_1/doc_1/de/pdf-a/fedlex-data-admin-ch-eli-dl-proj-2021-61-cons_1-doc_1-de-pdf-a.pdf, April 2021. Accessed at 2021-05-17.

[62] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter YA Ryan, Steve A Schneider, Vanessa Teague, Roland Wen, Zhe Xia, and Sriramkrishnan Srinivasan. Using Prêt à Voter in Victoria State Elections. *EVT/WOTE*, 2, 2012.

[63] Craig Burton, Chris Culnane, and Steve Schneider. Verifiable electronic voting in practice: the use of vvote in the victorian state election. *IEEE Security and Privacy*, 2016.

[64] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc,

Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, page 19, USA, 2010. USENIX Association.

[65] Peter Castenmiller and Kees Uijl. The use of'supporting software'in elections, The peculiar case of the Netherlands 2017. *Proceedings E-Vote*, pages 315–325, 2017.

[66] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1614–1625, 2016.

[67] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 177–182. Springer, 1988.

[68] David Chaum. Sure Vote: Technical Overview. In *Proceedings of the workshop on trustworthy elections (WOTE 2001)*, 2001.

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

[70] David Chaum, Richard T Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L Rivest, Peter YA Ryan, Emily Shen, Alan T Sherman, and Poorvi L Vora. Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE transactions on information forensics and security*, 4(4):611–627, 2009.

[71] David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida. *Electronic Elections: A Balancing Act*, pages 124–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

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

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

[74] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties

of voting schemes. In *Towards Trustworthy Elections*, pages 191–199. Springer, 2010.

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

[76] Bundesversammlung der Schweizerischen Eidgenossenschaft Claudio Zanetti. Kosten von E-Voting. https://www.parlament.ch/de/ratsbetrieb/suche-curia-vista/geschaeft?AffairId=20181083, December 2018. Accessed at 2021-11-26.

[77] Josh D Cohen and Michael J Fischer. *A robust and verifiable cryptographically secure election scheme*. Yale University. Department of Computer Science, 1985.

[78] New South Wales (AU) Electoral Commission. NSW Electoral Commission iVote and Swiss Post e-voting. https://www.elections.nsw.gov.au/About-us/Media-centre/News-media-releases/NSW-Electoral-Commission-iVote-and-Swiss-Post-e-vo, March 2019. Accessed at 2021-09-23.

[79] New South Wales (AU) Electoral Commission. NSW Electoral Commission iVote and Swiss Post e-voting update. https://www.elections.nsw.gov.au/About-us/Media-centre/News-media-releases/NSW-Electoral-Commission-iVote-and-Swiss-Post, March 2019. Accessed at 2021-09-23.

[80] Estonian National Electoral Committee. Comment on the article published in The Guardian. https://web.archive.org/web/20140617202431/http://vvk.ee/valimiste-korraldamine/vvk-uudised/vabariigi-valimiskomisjoni-vastulause-the-guardianis-ilmunud-artiklile/, May 2014. Accessed at 2021-10-29.

[81] nau.ch Conradin Zellweger. Fast Hälfte aller Wahlzettel in Schwamendingen ZH sind ungültig. https://www.nau.ch/politik/regional/fast-halfte-aller-wahlzettel-in-schwamendingen-zh-sind-ungultig-65305916, March 2018. Accessed at 2021-11-29.

[82] Véronique Cortier, Fabienne Eigner, Steve Kremer, Matteo Maffei, and Cyrille Wiedling. Type-based verification of electronic voting protocols. In *International Conference on Principles of Security and Trust*, pages 303–323. Springer, 2015.

[83] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Election verifiability for helios under weaker trust assumptions. In *European Symposium on Research in Computer Security*, pages 327–344. Springer, 2014.

[84] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798. IEEE, 2016.

[85] Véronique Cortier, Joseph Lallemand, and Bogdan Warinschi. Fifty shades of ballot privacy: Privacy against a malicious board. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 17–32. IEEE, 2020.

[86] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[87] Véronique Cortier and Cyrille Wiedling. A formal analysis of the Norwegian E-voting protocol. In *International Conference on Principles of Security and Trust*, pages 109–128. Springer, 2012.

[88] Véronique Cortier and Cyrille Wiedling. A formal analysis of the Norwegian E-voting protocol. *Journal of Computer Security*, 25(1):21–57, 2017.

[89] Chris Culnane, Mark Eldridge, Aleksander Essex, and Vanessa Teague. Trust implications of DDoS protection in online elections. In *International Joint Conference on Electronic Voting*, pages 127–145. Springer, 2017.

[90] Chris Culnane, Peter YA Ryan, Steve Schneider, and Vanessa Teague. vVote: a verifiable voting system. *ACM Transactions on Information and System Security (TISSEC)*, 18(1):1–30, 2015.

[91] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *European Symposium on Research in Computer Security*, pages 481–498. Springer, 2013.

[92] George Danezis and Microsoft Research Diaz, Claudia. Technical Report MSR-TR-2008-35: A survey of anonymous communication channels. report, Microsoft Research, 2008.

[93] Scytl David Galindo. Analysis of Cast-as-Intended and Counted-as-Recorded Verifiability for Scytl sVote Protocol using ProVerif. report, Scytl, November 2018. Accessed at 2021-06-17.

[94] Scytl David Galindo. sVote with Control Components Voting Protocol - Computational Proof of Complete Verifiability. report, Scytl, November 2018. Accessed at 2021-06-17.

[95] Denise Demirel, Maria Henning, Jeroen van de Graaf, Peter YA Ryan, and Johannes Buchmann. Prêt à voter providing everlasting privacy. In *International Conference on E-Voting and Identity*, pages 156–175. Springer, 2013.

[96] Denise Demirel, Jeroen Van De Graaf, and Roberto Samarone dos Santos Araújo. Improving Helios with Everlasting Privacy Towards the Public. *EVT/WOTE*, 12, 2012.

[97] Bundesversammlung der Schweizerischen Eidgenossenschaft. Internationaler Pakt über bürgerliche und politische Rechte. https://www.fedlex.admin.ch/eli/cc/1993/750_750_750/de, December 1966. Accessed at 2021-09-14.

[98] Bundesversammlung der Schweizerischen Eidgenossenschaft. Bundesgesetz über die politischen Rechte (BPR). https://www.fedlex.admin.ch/eli/cc/1978/688_688_688/de, December 1976. Accessed at 2021-05-17.

[99] swissinfo Domhnall O'Sullivan. Die Schweiz wird zum Briefwahl-Paradies. https://www.swissinfo.ch/ger/direkte-demokratie_die-schweiz-wird-zum-briefwahl-paradies/46073018, October 2020. Accessed at 2022-01-20.

[100] Ardita Driza Maurer. Report on the possible update of the Council of Europe Recommendation Rec (2004) 11 on legal, operational and technical standards for e-voting. report, Driza Maurer, Ardita, 2013.

[101] Ardita Driza-Maurer, Oliver Spycher, Geo Taglioni, and Anina Weber. E-voting for Swiss Abroad: A Joint Project between the Confederation and the Cantons. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[102] Eric Dubuis, Stephan Fischli, Rolf Haenni, Severin Hauser, Reto E Koenig, Philipp Locher, J Ritter, and Philémon von Bergen. Verifizierbare Internet-Wahlen an Schweizer Hochschulen mit UniVote. *INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt*, 2013.

[103] Eric Dubuis, Stephan Fischli, Rolf Haenni, Uwe Serdült, and Oliver Spycher. Selectio Helvetica: A verifiable remote e-voting system. Published at Bern University of Applied Sciences (BFH), 2011.

[104] David Duenas-Cid, Iuliia Krivonosova, Radu Serrano, Marlon Freire, and Robert Krimmer. Tripped at the Finishing Line: The Åland Islands Internet Voting Project. In Robert Krimmer, Melanie Volkamer, Bernhard Beckert, Ralf Küsters, Oksana Kulyk, David Duenas-Cid, and Mihkel Solvak, editors, *Electronic Voting*, pages 36–49, Cham, 2020. Springer International Publishing.

[105] Schweizerische Eidgenossenschaft. Bundesverfassung der Schweizerischen Eidgenossenschaft (BV). https://www.fedlex.admin.ch/eli/cc/1999/404/de, April 1999. Accessed at 2021-06-08.

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

[107] Chantal Enguehard and Camille Noûs. Some Things you may Want to Know about Electronic Voting in France. In *E-Vote-ID 2020*, Bregenz, Austria, Oct 2020.

[108] Berner Fachhochschule Eric Dubuis. E-Voting. https://gitlab.com/openchvote/cryptographic-protocol, May 2021.

[109] Aleks Essex, Jeremy Clark, and Carlisle Adams. Aperio: High integrity elections for developing countries. In *Towards Trustworthy Elections*, pages 388–401. Springer, 2010.

[110] Aleks Essex, Jeremy Clark, Richard Carback, and Stefan Popoveniuc. Punchscan in practice: An E2E election case study. In *Proceedings of Workshop on Trustworthy Elections*, 2007.

[111] Aleksander Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. Eperio: Mitigating Technical Complexity in Cryptographic Election Verification. *EVT/WOTE*, 13:116–135, 2010.

[112] République et canton de Genève. E-voting system - CHVote. https://republique-et-canton-de-geneve.github.io/chvote-1-0/index-en.html, January 2016.

[113] République et canton de Genève. Poursuite du développement de CHVote, le système de vote électronique genevois. https://www.ge.ch/actualite/poursuite-du-developpement-chvote-systeme-vote-electronique-genevois-23-03- March 2016. Accessed at 2021-05-21.

[114] République et canton de Genève. Genève met un terme au développement de sa plateforme de vote électronique CHVote. https://www.ge.ch/document/

point-presse-du-conseil-etat-du-28-novembre-2018, March 2018. Accessed at 2021-05-21.

[115] République et canton de Genève. Elections fédérales 2019: le canal de vote électronique ne sera pas proposé. https://www.ge.ch/document/point-presse-du-conseil-etat-du-19-juin-2019, June 2019. Accessed at 2021-05-21.

[116] République et canton de Genève. CHVote 1.0. https://github.com/republique-et-canton-de-geneve/chvote-1-0, May 2021.

[117] République et canton de Genève. CHVote 1.0: System Overview. https://github.com/republique-et-canton-de-geneve/chvote-1-0/blob/master/docs/system-overview.md, May 2021.

[118] République et canton de Genève. CHVote 2.0 Protocol PoC Code. https://github.com/republique-et-canton-de-geneve/chvote-protocol-poc, May 2021.

[119] République et canton de Genève. State of Geneva official release of the CHVote 2.0 project. https://chvote2.gitlab.io/, May 2021.

[120] République et canton de Neuchâtel. Neuchâtel, premier canton partenaire de la plateforme de vote électronique développée par La Poste. https://www.ne.ch/medias/Pages/150901_Neuch%C3%A2tel,-premier-canton-partenaire-de-la-plateforme-de-vote-%C3%A9lectronique-d%C3%A9velopp%C3%A9e-par-La-Poste.aspx, September 2015. Accessed at 2021-05-19.

[121] République et canton de Neuchâtel. Vote électronique. https://www.ne.ch/autorites/CHAN/CHAN/elections-votations/Pages/vote-electronique.aspx, September 2021. Accessed at 2021-09-28.

[122] International Organization for Standardization. Ergonomics of human-system interaction–part 11: Usability: Definitions and concepts. report, International Organization for Standardization, Geneva, CH, 2008.

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

[124] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.

[125] Bundesamt für Kommunikation. Strategie Digitale Schweiz. report, Bundesamt für Kommunikation, June 2020. Accessed at 2021-11-25.

[126] David Galindo, Sandra Guasch, and Jordi Puiggali. 2015 Neuchâtel's cast-as-intended verification mechanism. In *International Conference on E-Voting and Identity*, pages 3–18. Springer, 2015.

[127] Galois. *The Future of Voting*. US Vote Foundation, 2015.

[128] Ida Sofie Gebhardt Stenerud and Christian Bull. When reality comes knocking norwegian experiences with verifiable electronic voting. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[129] Bundesversammlung der Schweizerischen Eidgenossenschaft Genf. Entwicklung eines E-Voting-Systems durch den Bund oder die Kantone. https://www.parlament.ch/de/ratsbetrieb/suche-curia-vista/geschaeft?AffairId=20190312, September 2019. Accessed at 2021-09-29.

[130] Micha Germann. Internet voting increases expatriate voter turnout. *Government Information Quarterly*, 38(2):101560, Apr 2021.

[131] Micha Germann and Uwe Serdült. Internet voting and turnout: Evidence from Switzerland. *Electoral Studies*, 47:1–12, 2017.

[132] Kristian Gjøsteen. Analysis of an internet voting protocol. *IACR Cryptol. ePrint Arch.*, 2010:380, 2010.

[133] Kristian Gjøsteen. The Norwegian internet voting protocol. In *International Conference on E-Voting and Identity*, pages 1–18. Springer, 2011.

[134] Stephen N Goggin, Michael D Byrne, Juan E Gilbert, Gregory Rogers, and Jerome McClendon. Comparing the Auditability of Optical Scan, Voter Verified Paper Audit Trail (VVPAT) and Video (VVVAT) Ballot Systems. *EVT*, 8:1–7, 2008.

[135] Ben Goldsmith and Washington DC: National Democractic Institute Ruthrauff, Holly. Case Study Report on the Philippines 2010 Elections. report, Washington DC: National Democractic Institute, 2011. Accessed at 2021-09-12.

[136] Rolf Haenni, Eric Dubuis, Reto E Koenig, and Philipp Locher. CHVote: Sixteen Best Practices and Lessons Learned. In *International Joint Conference on Electronic Voting*, pages 95–111. Springer, 2020.

[137] Rolf Haenni, Reto Koenig, Stephan Fischli, and Eric Dubuis. TrustVote: A proposal for a hybrid e-voting system. First paper of new group at Bern University of Applied Sciences (BFH)., 2009.

[138] Rolf Haenni, Reto E Koenig, and Eric Dubuis. Cast-as-intended verification in electronic elections based on oblivious transfer. In *International Joint Conference on Electronic Voting*, pages 73–91. Springer, 2016.

[139] Rolf Haenni, Reto E Koenig, Philipp Locher, and Eric Dubuis. CHVote System Specification. *IACR Cryptol. ePrint Arch.*, 2017:325, 2017.

[140] Rolf Haenni and Oliver Spycher. Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys. *EVT/WOTE*, 11, 2011.

[141] Thomas Haines. Cronus: Everlasting Privacy with Audit and Cast. In *Nordic Conference on Secure IT Systems*, pages 53–68. Springer, 2019.

[142] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 644–660. IEEE, 2020.

[143] J Alex Halderman and Vanessa Teague. 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.

[144] Inside IT Hans Jörg Maron. E-Voting: Mindestens drei Kantone verlangen von der Post Schadenersatz. https://www.inside-it.ch/de/post/e-voting-mindestens-drei-kantone-verlangen-von-der-post-schadenersatz-20190708, July 2019. Accessed at 2021-10-31.

[145] Sven Heiberg, Kristjan Krips, and Jan Willemson. Planning the next steps for Estonian Internet voting. *E-Vote-ID 2020*, page 82, 2020.

[146] Sven Heiberg, Peeter Laud, and Jan Willemson. The application of i-voting for Estonian parliamentary elections of 2011. In *International Conference on E-Voting and Identity*, pages 208–223. Springer, 2011.

[147] Sven Heiberg, Helger Lipmaa, and Filip Van Laenen. On e-vote integrity in the case of malicious voter computers. In *European Symposium on Research in Computer Security*, pages 373–388. Springer, 2010.

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

[149] Sven Heiberg, Arnis Parsovs, and Jan Willemson. Log analysis of Estonian internet voting 2013–2014. In *International Conference on E-Voting and Identity*, pages 19–34. Springer, 2015.

[150] Sven Heiberg and Jan Willemson. Verifiable internet voting in Estonia. In *2014 6th international conference on electronic voting: Verifying the vote (evote)*, pages 1–8. IEEE, 2014.

[151] Jörg Helbach and Jörg Schwenk. Secure internet voting with code sheets. In *International Conference on E-Voting and Identity*, pages 166–177. Springer, 2007.

[152] Novobrief Jaime Novoa. Scytl: how a Barcelona-based company transformed the global elections industry in just 15 years. https://novobrief.com/scytl-pere-valles-interview/3479/, September 2015. Accessed at 2021-09-28.

[153] Rui Joaquim, Paulo Ferreira, and Carlos Ribeiro. EVIV: An end-to-end verifiable Internet voting system. *Computers & Security*, 32:170–191, 2013.

[154] Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira. Veryvote: A voter verifiable code voting system. In *International Conference on E-Voting and Identity*, pages 106–121. Springer, 2009.

[155] Hugo Jonker, Sjouke Mauw, and Jun Pang. Privacy and verifiability in voting systems: Methods, developments and trends. *Computer Science Review*, 10:1–30, 2013.

[156] Scytl Jordi Puiggali. Review of the attack described in the report "Faking an iVote decryption proof" by Vanessa Teague, Associate Professor, dated October 2, 2019. report, Scytl, November 2019.

[157] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *Workshop on Privacy in the Electronic Society*, pages 61–70, Nov 2005.

[158] Atte Juvonen. A framework for comparing the security of voting schemes. mastersthesis, University of Helsinki, 2019.

[159] Tyler Kaczmarek, John Wittrock, Richard Carback, Alex Florescu, Jan Rubio, Noel Runyan, Poorvi L Vora, and Filip Zagórski. Dispute resolution in accessible voting systems: The design and use of audiotegrity. In *International Conference on E-Voting and Identity*, pages 127–141. Springer, 2013.

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

[161] Reto E Koenig, Philipp Locher, and Rolf Haenni. Attacking the verification code mechanism in the norwegian internet voting system. In *International conference on e-voting and identity*, pages 76–92. Springer, 2013.

[162] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *European Symposium on Research in Computer Security*, pages 389–404. Springer, 2010.

[163] Robert Krimmer. Reviewing the Costs of Multichannel Elections: Estonian Parliamentary Elections 2019. *E-Vote-ID 2020*, page 356, 2020.

[164] Robert Krimmer, David Duenas-Cid, Iuliia Krivonosova, Priit Vinkel, and Arne Koitmae. How much does an e-Vote cost? Cost comparison per vote in multichannel elections in Estonia. In *International Joint Conference on electronic voting*, pages 117–131. Springer, 2018.

[165] Robert Krimmer, Andreas Ehringfeld, and Markus Traxl. The use of E-voting in the austrian federation of students elections 2009. In *4th International Conference on Electronic Voting 2010*. Gesellschaft für Informatik eV, 2010.

[166] Kristjan Krips and Jan Willemson. On practical aspects of coercion-resistant remote voting systems. In *International Joint Conference on Electronic Voting*, pages 216–232. Springer, 2019.

[167] Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. Extending helios towards private eligibility verifiability. In *International Conference on E-Voting and Identity*, pages 57–73. Springer, 2015.

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

[169] Ralf Küsters, Johannes Müller, Enrico Scapin, and Tomasz Truderung. sElect: A lightweight verifiable remote voting system. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 341–354. IEEE, 2016.

[170] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Improving and simplifying a variant of pret â voter. In *International Conference on E-Voting and Identity*, pages 37–53. Springer, 2009.

[171] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 526–535, 2010.

[172] Joseph Lallemand. *Vote électronique: définitions et techniques d'analyse*. phdthesis, Université de Stuttgart Christine Paulin Professeur, Université Paris-Sud, 2019.

[173] Lavanguardia Lalo Agustina. El juez abre la venta de Scytl y espera ofertas por la empresa hasta el 22 de junio. https://www.lavanguardia.com/economia/20200607/481657532094/scytl-subasta-unidad-productiva-venta-indra-liquidacion.html, June 2020. Accessed at 2021-09-28.

[174] Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. Ceci n'est pas une preuve. report, Lewis, Sarah Jamie and Pereira, Olivier and Teague, Vanessa, 2019-03-12. Accessed at 2021-10-11.

[175] Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. report, Lewis, Sarah Jamie and Pereira, Olivier and Teague, Vanessa, 2019-03-25. Accessed at 2021-10-11.

[176] Helger Lipmaa. Two Simple Code-Verification Voting Protocols. *IACR Cryptol. ePrint Arch.*, 2011:317, 2011.

[177] Philipp Locher and Rolf Haenni. Verifiable internet elections with everlasting privacy and minimal trust. In *International Conference on E-Voting and Identity*, pages 74–91. Springer, 2015.

[178] Philipp Locher and Rolf Haenni. Receipt-free remote electronic elections with everlasting privacy. *Annals of Telecommunications*, 71(7):323–336, 2016.

[179] Philipp Locher, Rolf Haenni, and Berner Fachhochschule Koenig, Reto E. Analysis of the Cryptographic Implementation of the Swiss Post Voting Protocol. report, Berner Fachhochschule, July 2019. Accessed at 2021-05-19.

[180] Philipp Locher, Rolf Haenni, and Reto E Koenig. Coercion-resistant internet voting with everlasting privacy. In *International Conference on Financial Cryptography and Data Security*, pages 161–175. Springer, 2016.

[181] Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*, pages 31–43. IEEE, 1997.

[182] David Lundin and Peter YA Ryan. Human readable paper verification of Pret a Voter. In *European Symposium on Research in Computer Security*, pages 379–395. Springer, 2008.

[183] Epp Maaten and Thad Hall. Improving the transparency of remote e-voting: The estonian experience. In *Electronic Voting 2008 (EVOTE08). 3rd International Conference on Electronic Voting 2008, Co-organized by Council of Europe, Gesellschaft für Informatik and EVoting. CC*. Gesellschaft für Informatik e. V., 2008.

[184] Ülle Madise and Tarvi Martens. E-voting in Estonia 2005. The first practice of country-wide binding Internet voting in the world. In *Electronic Voting 2006–2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting. CC*. Gesellschaft für Informatik eV, 2006.

[185] Ülle Madise and Priit Vinkel. Internet voting in Estonia: from constitutional debate to evaluation of experience over six elections. In *Regulating eTechnologies in the European Union*, pages 53–72. Springer, 2014.

[186] Randi Markussen, Lorena Ronquillo, and Carsten Schürmann. Trust in internet election observing the Norwegian decryption and counting ceremony. In *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, pages 1–8. IEEE, 2014.

[187] Karola Marky, Oksana Kulyk, Karen Renaud, and Melanie Volkamer. What did I really vote for? On the usability of verifiable e-voting schemes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[188] Verified Voting Matthew Caulfield. The Price of Voting: Today's Voting Machine Marketplace. report, Verified Voting, March 2021. Accessed at 2021-09-12.

[189] Ardita Driza Maurer. Updated European standards for e-voting. In *International Joint Conference on Electronic Voting*, pages 146–162. Springer, 2017.

[190] Silvio Micali and Michael O Rabin. Cryptography miracles, secure auctions, matching problem verification. *Communications of the ACM*, 57(2):85–93, 2014.

[191] Thomas Milic, Michele McArdle, Uwe Serdült, Andreas Glaser, Daniel Kübler, and Béatrice Ziegler. Haltungen und Bedürfnisse der Schweizer Bevölkerung zu E-Voting= Attitudes of Swiss citizens towards the generalisation of e-voting. *Studienberichte des Zentrums für Demokratie Aarau*, 9, 2016.

[192] Guillermo Lopez Mirau, Teresa Ovejero, and Julia Pomares. The implementation of e-voting in Latin America: the experience of salta, Argentina from a practitioner's perspective. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[193] Ester Moher, Jeremy Clark, and Aleksander Essex. Diffusion of voter responsibility: Potential failings in e2e voter receipt checking. {*USENIX*} *Journal of Election Technology and Systems ({JETS})*, 1:1–17, 2014.

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

[195] Tal Moran and Moni Naor. Split-ballot voting: everlasting privacy with distributed trust. *ACM Transactions on Information and System Security (TISSEC)*, 13(2):1–43, 2010.

[196] Moni Naor and Adi Shamir. Visual cryptography. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 1–12. Springer, 1994.

[197] C. Andrew Ne. Practical high certainty intent verification for encrypted votes. Accessed at 2021-10-23, 2004.

[198] André Silva Neto, Matheus Leite, Roberto Araújo, Marcelle Pereira Mota, Nelson Cruz Sampaio Neto, and Jacques Traoré. Usability considerations for coercion-resistant election systems. In *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*, pages 1–10, 2018.

[199] Stephan Neumann, Jurlind Budurushi, and Melanie Volkamer. Analysis of security and cryptographic approaches to provide secret and verifiable electronic voting. In *Design, development, and use of secure electronic voting systems*, pages 27–61. IGI Global, 2014.

[200] Stephan Neumann, C. Feier, M. Volkamer, and Reto E. Koenig. Towards A Practical JCJ / Civitas Implementation. In *GI-Jahrestagung*, 2013.

[201] Stephan Neumann, Christian Feier, Perihan Sahin, and Sebastian Fach. Pretty Understandable Democracy 2.0. *IACR Cryptol. ePrint Arch.*, 2014:625, 2014.

[202] Stephan Neumann and Melanie Volkamer. 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.

[203] Regjeringen Norway. Prop. 1 S (2009–2010): For budsjettåret 2010 under Kommunal- og regionaldepartementet. https://www.regjeringen.no/no/dokumenter/prop-1-s-20092010/id580467/, 2009. Accessed at 2021-09-30.

[204] Regjeringen Norway. Prosjektdirektiv for e-valg 2011-prosjektet. report, Regjeringen Norway, January 2011. Accessed at 2021-09-30.

[205] Regjeringen Norway. Presentasjon av prosjektet. https://www.regjeringen.no/no/dokumentarkiv/stoltenberg-ii/krd/tema-og-redaksjonelt-innhold/redaksjonelle-artikler/2010/presentasjonavprosjektet/id598565/, June 2013. Accessed at 2021-09-30.

[206] Regjeringen Norway. Systemarkitektur. report, Regjeringen Norway, June 2013. Accessed at 2021-09-30.

[207] Council of Europe. Recommendation CM/Rec(2004)11 of the Committee of Ministers to member States on Legal, operational and technical standards for E-Voting, September 2004.

[208] Council of Europe. Guidelines on the implementation of the provisions of Recommendation CM/Rec(2017)51 in standards for e-voting, June 2017.

[209] Council of Europe. Recommendation CM/Rec(2017)51 of the Committee of Ministers to member States on standards for e-voting, June 2017.

[210] Maina M Olembo, Steffen Bartsch, and Melanie Volkamer. Mental models of verifiability in voting. In *International Conference on E-Voting and Identity*, pages 142–155. Springer, 2013.

[211] Organisation for Security OSCE/ODIHR Election Assessment Mission, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Swiss confederation federal elections: 21. october 2007. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, April 2008.

[212] Organisation for Security OSCE/ODIHR Election Assessment Mission, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Swiss confederation federal elections: 23. october 2011. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, January 2012.

[213] Organisation for Security OSCE/ODIHR Election Assessment Mission, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Parlamentary elections 9 september 2011. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, December 2013.

[214] Organisation for Security OSCE/ODIHR Election Assessment Mission, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Swiss confederation federal elections: 18. october 2015. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, February 2016.

[215] Organisation for Security OSCE/ODIHR Election Expert Team, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Internet voting pilot project: Local government elections 12 september 2011. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, March 2012.

[216] Organisation for Security OSCE/ODIHR Election Expert Team, Cooperation in Europe's Office for Democratic Institutions, and Human Rights. Needs assessment mission report for Swiss confederation federal elections: 20. october 2019. report, Organisation for Security and Cooperation in Europe's Office for Democratic Institutions and Human Rights, July 2019.

[217] Bjarte M Østvold and Edvard K Karlsen. Public Review of E-Voting Source Code: Lessons learnt from E-Vote 2011. *Norsk informatikkonferanse*, 2012.

[218] Paragon. Paragon: Annual report. report, Paragon, 2020.

[219] electoral matters committee Parliament of Victoria. Inquiry into electronic voting. report, Parliament of Victoria, electoral matters committee, 2017.

[220] Thea Peacock, Peter YA Ryan, Steve Schneider, and Zhe Xia. Verifiable voting systems. In *Computer and information security handbook*, pages e293–e315. Elsevier, 2013.

[221] Olivier Pereira. Internet voting with Helios. In *Real-World Electronic Voting*, pages 293–324. Auerbach Publications, 2016.

[222] Olivier Pereira and Vanessa Teague. Report on the SwissPost-Scytl e-voting system, trusted-server version. report, Pereira, Olivier and Teague, Vanessa, July 2019. Accessed at 2021-05-19.

[223] Adrian Perrig, Pawel Szalachowski, Raphael M Reischuk, and Laurent Chuat. *SCION: a secure Internet architecture*. Springer, 2017.

[224] Julia Pomares, Ines Levin, R Michael Alvarez, Guillermo Lopez Mirau, and Teresa Ovejero. From piloting to roll-out: voting experience and trust in the first full e-election in argentina. In *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, pages 1–10. IEEE, 2014.

[225] Schweizerische Post. Post treibt eVoting voran. https://www.post.ch/de/ueber-uns/medien/medienmitteilungen/2015/post-treibt-evoting-voran, September 2015. Accessed at 2021-10-11.

[226] Schweizerische Post. Erfolgreiche Premiere für das E-Voting-System der Post im Kanton Freiburg. https://www.post.ch/de/ueber-uns/medien/medienmitteilungen/2016/erfolgreiche-premiere-fuer-das-e-voting-system-der-post-im-kanton-freiburg, November 2016. Accessed at 2021-10-11.

[227] Schweizerische Post. E-Voting der Post für 50 Prozent der Stimmbürger zertifiziert. https://www.post.ch/de/ueber-uns/medien/medienmitteilungen/2017/e-voting-der-post-fuer-50-prozent-der-stimmbuerger-zertifiziert, August 2017. Accessed at 2021-10-11.

[228] Schweizerische Post. Abschlussbericht Öffentlicher Intrusionstest E-Voting-System mit universeller Verifizierbarkeit. report, Schweizerische Post, June 2019. Accessed at 2021-05-19.

[229] Schweizerische Post. Die Post setzt ausschliesslich auf das neue System mit universeller Verifizierbarkeit. https://www.post.ch/de/ueber-uns/medien/medienmitteilungen/2019/die-post-setzt-ausschliesslich-auf-das-neue-system-mit-universeller-verifizierbark
July 2019. Accessed at 2021-05-21.

[230] Schweizerische Post. Fehler im Quellcode aufgedeckt und behoben: Technische Beschreibung des aufgedeckten Fehlers und Lösung. report, Schweizerische Post, March 2019. Accessed at 2021-05-21.

[231] Schweizerische Post. Urne nicht gehackt, Fehler im Quellcode – Post setzt ihr E-Voting-System befristet aus. https://www.post.ch/de/ueber-uns/medien/medienmitteilungen/2019/post-setzt-ihr-e-voting-system-befristet-aus, March 2019. Accessed at 2021-10-11.

[232] Schweizerische Post. Ein E-Voting-System für die Schweiz aus der Schweiz. https://www.evoting-blog.ch/de/pages/2020/ein-e-voting-system-fuer-die-schweiz-aus-der-schweiz, June 2020. Accessed at 2021-05-21.

[233] Schweizerische Post. Der Quellcode des zukünftigen E-Voting-Systems ist veröffentlicht. https://www.evoting-blog.ch/de/pages/2021/der-quellcode-des-zukuenftigen-e-voting-systems-ist-veroeffentlicht, September 2021. Accessed at 2021-10-22.

[234] Schweizerische Post. E-voting documentation. report, Schweizerische Post, October 2021. Accessed at 2021-10-21.

[235] Schweizerische Post. Protocol of the Swiss Post Voting System: Computational Proof of Complete Verifiability and Privacy. Version 0.9.10. report, Schweizerische Post, July 2021. Accessed at 2021-10-15.

[236] Schweizerische Post. Source code of the e-voting system. https://gitlab.com/swisspost-evoting/e-voting/e-voting, December 2021.

[237] Schweizerische Post. Vollständige Verifizierbarkeit und Open Source im zukünftigen E-Voting-System der Post. https://www.evoting-blog.ch/de/pages/2021/vollstaendige-verifizierbarkeit-und-open-source-im-zukuenftigen-e-voting-sy November 2021. Accessed at 2021-12-01.

[238] Michael O Rabin and Ronald L Rivest. Efficient end to end verifiable electronic voting employing split value representations. *EVOTE 2014*, 2014.

[239] Michael O Rabin, Rocco A Servedio, and Christopher Thorpe. Highly efficient secrecy-preserving proofs of correctness of computations and applications. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 63–76. IEEE, 2007.

[240] Joël Reber. E-Voting in der Schweiz: Eine vergleichende Analyse der Entwicklungsstände und Haltungen in den Kantonen. mastersthesis, Université de Lausanne, 2018.

[241] Kantons Zürich Regierungsrat. 924. Anfrage (Grundsatzfragen zu E-Voting) KR-Nr. 154/2018. report, Regierungsrat, Kantons Zürich, September 2018.

[242] Markus Reiners. Rahmenbedingungen eines E-Voting in Deutschland, der Schweiz und Österreich. report, Markus Reiners, June 2021.

[243] Reuters. France drops electronic voting for citizens abroad over cybersecurity fears. https://www.reuters.com/article/idUSKBN16D233, March 2017. Accessed at 2021-10-20.

[244] Ronald L Rivest. Electronic voting. In *Financial Cryptography*, volume 1, pages 243–268. Citeseer, 2001.

[245] Ronald L Rivest. The threeballot voting system. In *VTP Working Paper Series;56*. Caltech/MIT Voting Technology Project, 2006-10-01.

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

[247] Ronald L Rivest and Warren D Smith. Three voting protocols: ThreeBallot, VAV, and Twin. *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*, 2007.

[248] Ronald L Rivest and Madars Virza. Software independence revisited. In *Real-World Electronic Voting*, pages 19–34. Auerbach Publications, 2016.

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

[250] Adrià Rodríguez-Pérez. My Vote, My (Personal) Data: Remote Electronic Voting and the General Data Protection Regulation. In *Electronic Voting*, pages 167–182. Springer, Cham, Switzerland, Sep 2020.

[251] Berner Fachhochschule Rolf Hänni. Swiss Post Public Intrusion Test: Undetectable Attack Against Vote Integrity and Secrecy. report, Berner Fachhochschule, 2019. Accessed at 2021-05-19.

[252] Peter B Rønne, Peter YA Ryan, and Marie-Laure Zollinger. Electryo, in-person voting with transparent voter verifiability and eligibility verifiability. *arXiv preprint arXiv:2105.14783*, 2021.

[253] Peter YA Ryan. Prêt à Voter with Paillier encryption. *Mathematical and Computer Modelling*, 48(9-10):1646–1662, 2008.

[254] Peter YA Ryan. Prêt à Voter with Confirmation Codes. *EVT/WOTE*, 11, 2011.

[255] Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *International Conference on Financial Cryptography and Data Security*, pages 176–192. Springer, 2016.

[256] Peter YA Ryan, Steve Schneider, and Vanessa Teague. Prêt à Voter—The Evolution of the Species. In *Real-World Electronic Voting*, pages 325–358. Auerbach Publications, 2016.

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

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

[259] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 393–403. Springer, 1995.

[260] BZ Basel Samuel Hufschmid. Basler Parlament gibt klein bei: E-Voting soll weiterhin möglich bleiben. https://www.bzbasel.ch/basel/basel-stadt/basler-parlament-gibt-klein-bei-e-voting-soll-weiterhin-moglich-bleiben-ld1372798, June 2019. Accessed at 2021-10-22.

[261] BZ Basel Samuel Hufschmid. E-Voting-Notbremse kostet Basel-Stadt mehrere Millionen Franken. https://www.bzbasel.ch/basel/basel-stadt/e-voting-notbremse-kostet-basel-stadt-mehrere-millionen-franken-ld.1346479, February 2019. Accessed at 2021-10-22.

[262] Scytl. sVote Voting with Control Components Protocol - Cryptographic proof of Privacy. report, Scytl, November 2018. Accessed at 2021-06-17.

[263] Scytl. Swiss Online Voting Protocol. report, Scytl, 2018.

[264] Scytl. Analysis of Ballot Privacy Property for Scytl sVote Protocol using ProVerif. report, Scytl, January 2019. Accessed at 2021-06-17.

[265] République et canton de Genève Secrétariat du Grand Conseil. PL 11867 - ouvrant un crédit d'investissement de 4 740 000 F pour la

réalisation du vote électronique de 2e génération. https://ge.ch/grandconseil/data/texte/PL11867.pdf, September 2016. Accessed at 2021-08-28.

[266] Fateme Shirazi, Stephan Neumann, Ines Ciolacu, and Melanie Volkamer. Robust electronic voting: Introducing robustness in civitas. In *2011 International Workshop on Requirements Engineering for Electronic Voting Systems*, pages 47–55. IEEE, 2011.

[267] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *European Symposium on Research in Computer Security*, pages 463–480. Springer, 2013.

[268] Ben Smyth, Mark Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, pages 146–163. Springer, 2010.

[269] Mihkel Solvak. Does vote verification work: usage and impact of confidence building technology in Internet voting. In *International Joint Conference on Electronic Voting*, pages 213–228. Springer, 2020.

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

[271] Oliver Spycher, Reto Koenig, Rolf Haenni, and Michael Schläpfer. 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.

[272] Oliver Spycher, Reto Koenig, Rolf Haenni, and Michael Schläpfer. Achieving meaningful efficiency in coercion-resistant, verifiable internet voting. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV, 2012.

[273] Oliver Spycher, Melanie Volkamer, and Reto Koenig. Transparency and technical measures to establish trust in norwegian internet voting. In *International Conference on E-Voting and Identity*, pages 19–35. Springer, 2011.

[274] Kanton Zürich Statistisches Amt, Direktion der Justiz und des Innern. Evaluation der E-Voting Testphase im Kanton Zürich 2008-2011. report, Statistisches Amt, Direktion der Justiz und des Innern, Kanton Zürich, November 2011. Accessed at 2021-05-19.

[275] Aargauer Zeitung Sven Altermatt. Keine Beobachter geschickt: Schweizer Wahlen ohne Kontrolleure, October 2019.

[276] Vanessa Teague and Patrick Keyzer. Electronic Australian Elections: Verifiability of Accuracy is a Design Goal, which Must be Mandated by Law and Deliberately Designed into Electronic Electoral Processes. *Law in Context. A Socio-legal Journal*, 37(1):1–24, 2020.

[277] Johannes Mueller Thomas Haines. How not to VoteAgain: Pitfalls of Scalable Coercion-Resistant E-Voting. *IACR Cryptol. ePrint Arch.*, 2020:1406, 2020.

[278] Eidgenössisches Justiz und Polizeidepartement EJPD. Elektronische Identität: das E-ID-Gesetz. https://www.ejpd.admin.ch/ejpd/de/home/themen/abstimmungen/bgeid.html, March 2021. Accessed at 2022-01-29.

[279] Taavi Unt, Mihkel Solvak, and Kristjan Vassil. Does Internet voting make elections less social? Group voting patterns in Estonian e-voting log files (2013–2015). *PloS one*, 12(5):e0177864, 2017.

[280] The University of Melbourne Vanessa Teague. Faking an iVote decryption proof: Why the decryption proof flaw identified in the Swiss Post system affects the iVote system too. report, The University of Melbourne, November 2019.

[281] Kristjan Vassil and Till Weber. A bottleneck model of e-voting: Why technology fails to boost turnout. *New media & society*, 13(8):1336–1354, 2011.

[282] Priit Vinkel and Robert Krimmer. The how and why to internet voting an attempt to explain E-Stonia. In *International joint conference on electronic voting*, pages 178–191. Springer, 2016.

[283] Melanie Volkamer, Jurlind Budurushi, and Denise Demirel. Vote casting device with VV-SV-PAT for elections with complicated ballot papers. In *2011 International Workshop on Requirements Engineering for Electronic Voting Systems*, pages 1–8. IEEE, 2011.

[284] Melanie Volkamer and Rüdiger Grimm. Multiple casts in online voting: Analyzing chances. In *Electronic Voting 2006–2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting. CC*. Gesellschaft für Informatik eV, 2006.

[285] Scytl Secure Electronic Voting. Implementing a public security scrutiny of an online voting system: the Swiss experience. In *International Joint Conference on Electronic Voting*, pages 311–329. Springer, 2019.

[286] Aaron Wilson. Modeling Requirements Conflicts in Secret Ballot Elections. *E-Vote-ID 2019*, page 171, 2019.

[287] Filip Zagórski, Richard T Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *International Conference on Applied Cryptography and Network Security*, pages 441–457. Springer, 2013.

[288] Marie-Laure Zollinger, Verena Distler, Peter B Roenne, Peter YA Ryan, Carine Lallemand, and Vincent Koenig. User Experience Design for E-Voting: How mental models align with security mechanisms. *arXiv preprint arXiv:2105.14901*, 2021.

[289] Regierungsrat Zürich. Regierungsratsbeschluss Nr. 61/2016: E-Voting, Auflösungsvereinbarung Consortium Vote électronique, Zustimmung. https://www.zh.ch/de/politik-staat/gesetze-beschluesse/beschluesse-des-regierungsrates/rrb/regierungsratsbeschluss-61-2016.html, January 2016. Accessed at 2021-10-23.

[290] Consortium Vote électronique. Deutlicher Rückschlag für E-Voting. https://www.zh.ch/content/dam/zhweb/bilder-dokumente/themen/politik-staat/wahlen-abstimmungen/abstimmungen/mm_consortium_ablehnung_gesuch_evoting_nrw.pdf, August 2015. Accessed at 2021-06-08.