Chapter 8

An Overview of End-to-End Verifiable Voting Systems

Syed Taha Ali
School of Computing Science
Newcastle University, UK
taha.ali@newcastle.ac.uk

Judy Murray
School of Geography, Politics and Sociology
Newcastle University, UK
judy.murray@newcastle.ac.uk

CONTENTS

8.1 Introduction

The use of cryptography to secure elections was first suggested by Chaum in 1981 in a highly influential paper on anonymous communications [153]. Chaum described new cryptographic primitives, or building blocks, and proposed various applications for these, including the possibility of conducting remote electronic elections. This idea proved popular in the academic research community, and, over the next two decades, a multitude of voting systems appeared in the literature, a handful of which resulted in actual prototype solutions [179] [306]. However, research in this domain was, for the most part, confined to academia and progressed independently of developments in real-world election technology.

In the last 15 years however the situation has dramatically altered. Research in securing elections has received a significant boost for two main reasons: First, the gridlock in the US presidential election of 2000 between George W. Bush and Al

Gore cast a spotlight on the deficiencies of the US voting infrastructure where voters faced considerable difficulties in casting votes due to confusing ballot designs and punch card voting machines [363]. This flurry of national attention led to passage of the Help America Vote Act (HAVA) in 2002 which, among other provisions, allocated generous funding for research to improve voting technology [162].

Second, multiple investigations in recent years have highlighted glaring reliability and security issues in electronic voting machines which render them vulnerable to hackers and compromise the integrity of elections [351] [232] [578] [535]. Indeed, there have been numerous documented instances of voting machines inexplicably malfunctioning during live elections, flipping candidate votes, or randomly adding and subtracting candidate votes. Since then, academic research interest in securing elections has soared, and prompted the formation of dedicated conferences and journals and research organizations [1] [18].

This renewed focus has led to the development of a new type of voting system with **end-to-end verifiable (E2E) security** properties. This highly promising advance, due to individual efforts by Chaum [143] and Neff [418] in 2004, harmonizes theoretical research innovation with the ground realities of election administration to provide strong guarantees on the integrity of elections. We describe next a typical usage scenario which encapsulates the spirit of E2E voting systems.

On election day, our voter, Alice goes to the polling station to cast her vote. She identifies herself to polling staff as a legitimate voter and is guided to a private booth with a voting machine, where she chooses her candidate on the touchscreen. The machine issues her a *receipt*, which is essentially a cryptographically masked copy of her vote. In the booth, Alice can also verify, either visually, or by challenging the machine, that her vote was **cast as intended**, i.e., the cryptographic obfuscation does indeed represent her chosen candidate and not another. After polls close, the system posts copies of all receipts on a public bulletin board or website where Alice confirms that her vote has been **recorded as cast**, i.e., her receipt is included in the lot and it matches the physical copy she has in her hand. In the final step, all receipts on the website are processed in a series of cryptographic computations to yield the election result. The algorithms and parameters for these operations are specified on the website, and any technically-minded person, including Alice herself, can therefore verify that her own vote was **tallied as recorded** and that the tally is indeed correct.

This scenario is starkly different from the current state of real-world elections where Alice has to implicitly trust the voting system and its administrators for the credibility of the election. Dishonest polling staff have been known to manipulate election results. Furthermore, the internal operations of voting machines are opaque to Alice and she has no assurance that the machine is actually doing what it is supposed to.[1] E2E voting systems, on the other hand, preclude trust in personnel and machines and make the voter herself an active participant in auditing the election at

[1] This realization was fundamental to the decision of the Federal Constitutional Court of Germany in 2009 declaring electronic voting as "unconstitutional" and thereby marking Germany's return to paper-based voting [37].

every step and certifying its result. If a voting machine loses Alice's vote or switches it to another candidate, her receipt on the website will reflect this change and Alice can file a complaint, using her own physical copy as evidence. If polling staff tamper with the final tally, any third party running the tallying or verification algorithm on their own computers will pick up on it. Furthermore, since Alice's receipt is an obfuscation of her choice, she cannot use it to convince a third party of how she voted, thereby thwarting vote-selling and coercion.

We provide here a comprehensive high-level introduction to the field of E2E voting. The writing is aimed at the layman with little knowledge of cryptography and attempts to communicate a clear and intuitive appreciation of E2E voting systems. This chapter is organized as follows: we introduce security properties of voting systems in Section 8.2. In Sections 8.3–8.4, we summarize the workings of some twenty of the most influential E2E voting systems, classified into four distinct categories, as per their reliance on cryptography (cryptographic and non-cryptographic systems), ballot format (physical and electronic ballots) and mode of voting (precinct-based and remote voting). This is followed by a discussion of open challenges to mainstream deployment of E2E voting systems in Section 8.5. We conclude in Section 8.6.

8.2 Security Properties of Voting Systems

Here we describe key security properties characterizing E2E voting systems. Many of these properties are intimately related whereas others are in direct conflict. The merits of a system are typically assessed on the basis of what properties it provides and how successfully their inherent conflicts are harmonized within the system.

8.2.1 Vote Privacy

The privacy of the vote is widely recognized as a fundamental human right and is enshrined in Article 21 of the Universal Declaration of Human Rights [67]. The rationale behind this, dating back to ancient Greece and Rome [286], is that if outside parties become privy to a voter's choice, it opens the door to bribery and intimidation, ultimately corrupting the electoral process. Vote-buying was not uncommon in the Western world up until the last century [547] and still persists in some developing countries [323] [265]. Likewise, voters may be intimidated by criminals, local politicians, or even family members, to vote a certain way. These fears directly motivated the notion of the *secret ballot*, which is typically implemented nowadays by providing the voter a private voting booth at the polling station. Her ballot bears no distinguishing marks and her vote is cast in the ballot box where it mixes with all the other votes, making it very difficult, if not impossible, to ascertain her choice.

In contrast, voting systems in the early research literature (such as [161] [91] [83] [140]) explicitly maintained the voter-ballot linkage by issuing the voter a re-

ceipt enabling her to track her vote on a public bulletin board. In 1994, Benaloh and Tuinstra [89] (and independently, Niemi and Renvall [415]) pointed out that this strategy allowed the voter to prove her vote to a third party after the election, thereby facilitating vote-buying and coercion. The authors introduced the notion of *receipt-freeness* which proved influential and several voting systems that followed (such as [89] [415] [508] [437] [438] [309]) dispensed with receipts entirely, focussing instead on ensuring transparency and integrity of the tallying operations. However, in 2004, Chaum [143] reintroduced the use of receipts, with the important distinction that the contents of the receipt are cryptographically masked, thereby maintaining the voter's privacy.

Juels, Catalano and Jakobsson [335] in 2005 argued that a coercer may yet influence a voter's choice without explicit knowledge of her vote, and described three such modes of coercion: the coercer may prevent the voter from voting, he may appropriate her voting credentials, or force her to vote for a candidate at random. The exact difference between receipt-freeness and coercion resistance is a subtle one [194]: in receipt-freeness, the coercer is assumed to be restricted to observing the election and using evidence provided by a cooperating voter. In the case of coercion resistance, the coercer is more powerful, and may craft specific votes for the voter to cast or even interact with her in some way while she is voting. Juels et al. proposed a solution to defend against these threats which we describe in Section 8.3.3.2.

Researchers have therefore progressively refined notions of vote privacy over the years into the following key properties:

- **Ballot Secrecy**: the voting system must not reveal who the voter voted for.
- **Receipt-Freeness**: the voting system should not give the voter any evidence to prove to a third party how she voted.
- **Coercion-Resistance**: the voter should be able to cast a vote for her intended choice even while appearing to cooperate with a coercer.

Intuitively, we see that each property encapsulates the previous one: coercion-resistance implies receipt-freeness which in turn implies ballot secrecy.

8.2.2 Vote Verifiability

In real-world elections, the voter has to implicitly trust voting machines and polling staff for the integrity of the elections. Typically there are ancillary processes in place to improve voter confidence in results, such as exit polls, random audits, and opening the tallying process to the public. On the other hand, voting systems in the research literature attempt to minimize the voter's dependence on personnel and machines, and use cryptography to provide ironclad guarantees on election integrity. Sako and Kilian [508] distinguished between two forms of verifiability:

- **Individual verifiability**: a voter can verify that her vote is included in the set of all cast votes.
- **Universal verifiability**: an observer can verify that the tally has been correctly computed from the set of all cast votes.

E2E voting systems recast the notion of verifiability in terms of three core steps:

- **Cast-as-intended**: the voter can verify the voting system correctly marked her candidate choice on the ballot.
- **Recorded-as-cast**: the voter can verify that her vote was correctly recorded by the voting system.
- **Tallied-as-recorded**: the voter can verify that her vote was counted as recorded.

This translates as follows: the voter first confirms the system has correctly encrypted her vote. She then tracks her vote on the bulletin board using her receipt and confirms that it is correctly recorded. Integrity of the result is ensured by rigorously auditing the tallying process and requiring the system to publish cryptographic proofs of correct operation. This three-step verification therefore covers the entire life cycle of the vote, and the voter can be confident that if there is any tampering or breakdown in the system it will be discovered in one of the checks.

These two conceptions of verifiability are also linked: in most E2E voting systems, cast-as-intended and recorded-as-cast checks are verified by the voter, i.e., together they provide individual verifiability, whereas the tallied-as-recorded step may be undertaken by voters and observers alike, i.e., it provides universal verifiability.

8.2.3 Other Properties

We list here certain additional properties also vitally important in voting systems:

- **Eligibility verifiability**: an observer can verify that each vote in the set of all cast votes was cast by an eligible voter.
- **Accountability**: in case vote verification fails at some stage, possibly due to error or fraud, the voter should be able to conclusively prove that it failed to the relevant authorities, without compromising the secrecy of her ballot.
- **Robustness**: the system should be robust to a certain degree of malfunction or corruption and still deliver correct results. A small number of misbehaving voters or system failures should not disrupt the election.
- **Usability**: the system should enable voters to cast their votes easily and effectively.
- **Accessability**: the system provides equal opportunity for access and participation (including guarantees on vote privacy and verifiability) to voters with disabilities.

8.2.4 Conflicts and Challenges

Several of the key properties described thus far conflict with each other in subtle ways, resulting in a variety of technical and legal challenges. For instance, as noted in the discussion on receipt-freeness, vote privacy clashes with vote verifiability. If a voter can successfully prove her vote to a third party outside the polling station, she could easily sell her vote or be coerced into voting for a certain candidate. A detailed treatment of this tension between verifiability and privacy in voting systems may be found in [334] and [407].

Similarly, there is a tension between vote verifiability and usability. Requiring voters to verify their vote negatively impacts usability by adding extra steps to the process, which may prove confusing for the voter [339] [40] [402]. Furthermore, experimental trials have noted that only a very small percentage of voters actually verify their vote [133] [395], which may critically undermine the security guarantees of these voting systems. We discuss these issues in greater detail in Section 8.5.2.

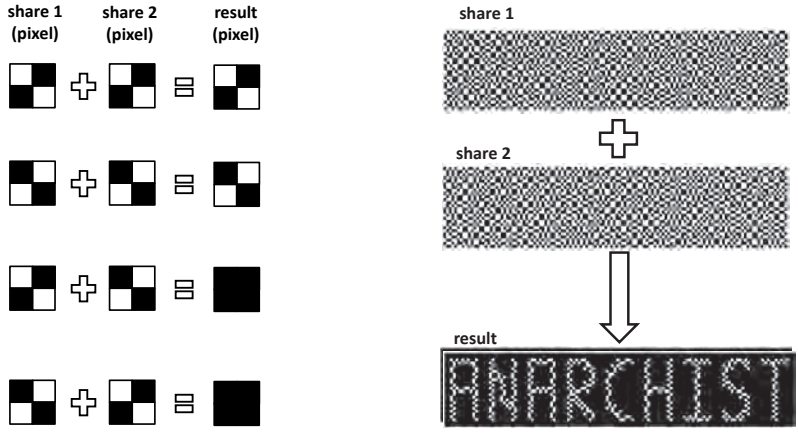
Accessibility also conflicts with vote privacy. Adapting voting systems to provide audio/visual aids or human assistance for voters with disabilities may create situations where the voter's candidate choice is revealed to a third party. Likewise, deploying voting systems for remote scenarios, such as postal or Internet voting, while significantly more convenient compared to in-person voting at polling stations, results in a marked deterioration in vote privacy [485]. In her home the voter is not guaranteed privacy and is also vulnerable to coercion.

Faced with such situations, designers of voting systems typically incorporate technological remedies or procedural safeguards into the system, prioritize certain security requirements over others or perhaps even focus on satisfying certain properties in a weaker form. We encounter several such examples in the following sections.

8.3 Cryptographic E2E Voting Systems

In this section, we describe notable E2E voting systems which rely on cryptography to guarantee E2E verifiability. These systems are further categorized depending on the ballot format they support: *physical ballots* (i.e., paper ballots) as opposed to *electronic ballots*, and their mode of deployment, for *precinct-based voting* (i.e., in-person voting at polling stations) versus *remote voting*.

There is considerable cross-fertilization across categories. Many systems incorporate common cryptographic primitives and procedural innovations. Furthermore, several systems share a common lineage and design philosophy which we have attempted to highlight in our presentation. We have also included certain systems which provide only a subset of E2E security properties, i.e., they are not strictly E2E verifiable, but nevertheless make an important contribution to the field.



(a) Construction of *white/black* pixels (b) Reconstruction of vote from shares

Figure 8.1: Generating a Votegrity receipt (with a vote for *Anarchist*).

8.3.1 Precinct-Based Voting with Physical Ballots

This category comprises precinct-based systems in which the tally is computed from receipts of paper ballots which voters mark by hand or using a machine. These systems are directly descended from Votegrity, and prominent examples include Scantegrity and Prêt à Voter, which have been deployed in politically binding elections.

8.3.1.1 Votegrity

Votegrity [143] was invented by Chaum in 2004 and is one of the very first and most influential E2E voting systems.[2] Votegrity uses visual cryptography to render the verification process more intelligible to the layman.

In **visual cryptography** [405], an image is split into multiple shares, such that individual shares do not yield any meaningful information about the original image and appear to contain random information. However, when the shares are overlaid the original image is reconstructed. As depicted in Figure 8.1(a) for the case of two shares, each pixel symbol is subdivided into four sub-pixels. When the shares are superimposed, identical pixels on both layers result in a semi-transparent or *white* pixel, whereas, if the two pixels are different, the resulting pixel will be opaque or *black*. Individual pixels on a layer reveal no information about the final result.

Vote processing in Votegrity is undertaken by a group of election trustees.[3]

[2]Several systems in the literature predating Votegrity offer what are arguably E2E security properties but these are primarily theoretical protocols based on abstract and idealized assumptions. Votegrity is the first E2E verifiable voting system to address the human complexities and practical realities of elections.

[3]The term *election trustee* in this chapter is taken to refer specifically to election personnel who handle cryptographic keys used in the voting systems.

Public-key cryptography is employed to maintain voter privacy and integrity of the tally. In this paradigm, each trustee possesses two keys, a *public* key he uses for data encryption and which is publicly posted, for example, on the election website or in a directory, and a *private* key used for decryption, which he keeps strictly private. These two keys are mutually related such that any entity can use the trustee's public key to encrypt data but only the trustee himself can decrypt it since he alone owns the corresponding private key. Many security protocols, including voting systems, specify multiple trustees to diffuse responsibility and trust. In this case, a system's security, may only be compromised if all trustees secretly collude. Trustees are therefore typically chosen such that they are mutually distrusting, i.e., they may belong to competing political parties and include activists and members of citizen groups.

On election day, Alice casts her vote at her local polling station. She enters her candidate choice on a voting machine, and presses a button, and a printer attached to the machine prints patterns on two strips of paper using visual cryptography. These strips are presented superimposed under a custom viewfinder, and the voter's choice is clearly visible. The example in Figure 8.1(b) depicts a vote for *Anarchist*.

Pixels on both strips are generated using a pseudorandom function, i.e., the patterns on the individual images appear random to an observer but are actually generated in a deterministic manner.[4] The voting machine also imprints a serial number and some validating information on both strips. Alice randomly chooses one of the strips as a receipt to take home. The machine shreds the other strip, issues the chosen strip to Alice, and saves a digital image of it to memory.

After polls close, the system publishes all saved voter receipts on a public bulletin board or website. Alice can locate her receipt using the serial number and verify that the digital image and validating information matches with her physical receipt. If there is any discrepancy, i.e., if the online receipt is missing or if the two versions do not match, she can initiate an inquiry with election authorities.

Votegrity's key breakthrough is this new role of the receipt. The printed information includes character strings which appear random to an observer but are actually a digital encoding of data which enables election trustees to fully reconstruct both original strips, and thereby the vote itself. When the voting machine generates the strips, it encrypts a digital copy of the visual patterns on both using the trustees' public keys, and this encryption, or *ciphertext*, is then printed on the receipt. The encryption may be understood with the analogy of an onion, where the vote is successively encapsulated in multiple skins or layers of encryption, each corresponding to a particular trustee. In the public-key paradigm only trustees can decrypt a vote by applying their private keys individually to remove corresponding layers of encryption.

The system decrypts cast votes in a privacy-preserving and verifiable manner using **decryption mixes**. A mix, invented by Chaum [153], is essentially a *permutation box* which accepts multiple data items, removes a layer of encryption, shuffles the items, and outputs them. A **mixnet** consists of multiple such mixes connected to-

[4]Details on construction of the patterns can be found in [143] and [152].

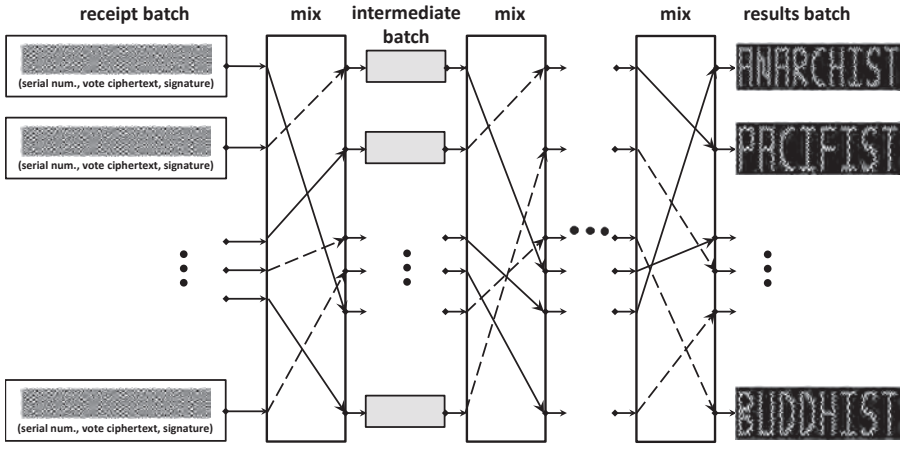


Figure 8.2: Votegrity decryption, mixing and audit process.

gether in cascading order such that the outputs of one feed into the next, as depicted in Figure 8.2. A mixnet effectively obscures the link between an input and an output and is a key ingredient in various anonymous communications protocols. Each mix is operated by an individual election trustee.

All receipts on the bulletin board are fed into the mixnet. The first mix strips the serial numbers from the receipts in addition to decryption and shuffling. Each mix removes a layer of encryption from the ciphertexts, figuratively peeling the onion, using the private key of the trustee operating the mix. The final result is a set of fully decrypted, or *plaintext*, votes stripped of any identifying information regarding the voter. These votes are then tallied in a straightforward manner.

Each mix processes its batch privately. Alice can be assured her vote is secret as revealing it would require that all the trustees collude to track her vote across each mix. However, there is still the issue of integrity: Alice cannot be sure the mixes are correctly processing the receipts and not altering them in transit. Chaum suggests election authorities publicly audit the mixes using a technique known as **randomized partial checking** [321]. The key idea is very simple: after tallying is concluded, each mix is forced to reveal a randomly chosen selection of half its inputs and outputs. These are posted on the bulletin board so that any observer can verify that the mix decrypted and mixed the items correctly. If a mix tampers with a single vote, there is a 50% chance it will get caught in a random check. The odds of being detected increase with every additional vote the mix manipulates, and would be close to 100% if a mix manipulates enough votes to influence a large-scale election.

However, while it is essential that mixes be subjected to this check, it is also vital that no receipt on the bulletin board be traced through the mixes to a plaintext vote. This is accomplished by selecting input-output links in an exclusory manner such

that no *end-to-end* path across the mixnet is revealed. For example, considering the first two mixes in Figure 8.2, using dashed lines to denote unmasked links, we see that paths revealed in one mix are kept strictly private in the adjacent mix.

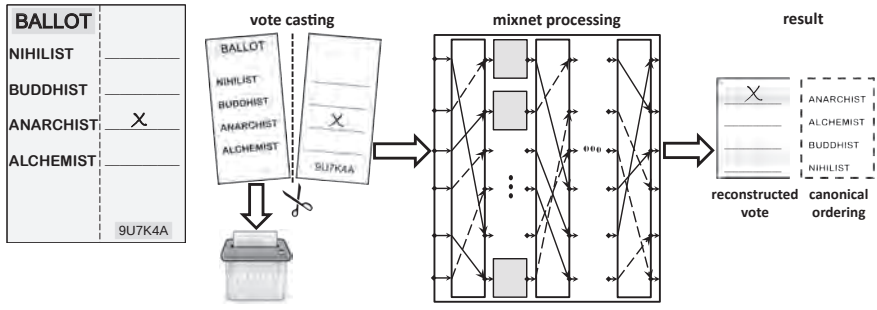
Now that the entire system has been described, we discuss the security properties it offers: when Alice casts her vote, she can visually verify that the printer correctly printed her candidate choice. Furthermore, each strip is digitally signed by the voting machine. **Digital signatures** are cryptographic equivalents of real signatures and have legal standing in several countries. In this case, the voting machine itself has a public/private key pair and their role is reversed: the machine's private key is used to mathematically compute a "signature" on the information on the receipt. This is a string of data which any observer can easily verify using the voting machine's public key (available via the bulletin board or in a public directory). Successful verification is undeniable proof of origin, as only the machine possesses the private key responsible for the signature. In case of a dispute, a digital signature is thus non-repudiable proof that the receipt was issued by a legitimate voting machine.

By digitally signing each strip, the machine may therefore be considered to have *committed* itself to both strips. Alice then chooses one layer at random to take home. If the machine were to cheat and put fraudulent information on single strips for a significant number of voters, it would be detected with very high probability.

Outside the polling station, Alice can verify, using the public keys of the trustees, that the ciphertext on the receipt is indeed a correct encryption of the visual pattern on her receipt. This gives her high assurance that her vote was cast as she intended. She can ensure the system correctly recorded her vote by verifying her physical receipt on the bulletin board. When mixing and tallying are undertaken, the inputs and outputs of every mix are publicly posted on the bulletin board. Any observer, including Alice, can monitor the random checks on the mixnets and confirm the results. Randomized partial checking is well-suited to audit elections as it does not rely on complex cryptography and is therefore efficient and relatively easier for the voter to appreciate. Any observer can verify the final tally for themselves by adding up all the decrypted votes. Furthermore, none of the information on the receipt or the bulletin board leaks any knowledge of the contents of Alice's vote which a third party may exploit to bribe or intimidate her. The visual image on her receipt is incomprehensible without the second strip which the voting machine shredded during the vote-casting phase, and theoretically it could represent a vote for *any* candidate.

In short, Alice can now audit the election at every key stage herself and need not trust election authorities for election integrity. Furthermore, election verifiability is independent of underlying infrastructure such as voting machines or vendor software. Technically-minded voters can create (and distribute) software to perform the necessary verifications using the information on their receipts and the bulletin board.

Votegrity has proved immensely influential and inaugurated new directions for research in secure elections. Several systems that followed borrow the basic receipt-and-mixnet template and improve on practical aspects. We consider these next.



(a) Ballot (b) Vote casting and tallying

Figure 8.3: Voting with Prêt à Voter(casting a vote for *Anarchist*).

8.3.1.2 Prêt à Voter

Prêt à Voter [499], initially proposed by Ryan in 2004 as a variant of Votegrity [500], was later developed into an independent system by Chaum, Ryan and Schneider [151]. Prêt à Voter is described in detail in Chapter 12. We provide a brief overview.

Prêt à Voter replaces Votegrity's visual cryptography element with a paper ballot and randomized candidate ordering. The ballot, depicted in Figure 8.3(a), is detachable into two halves along the perforated line in the middle. The left half lists candidates in randomized order. The right half has corresponding marking spaces and a random-looking string of alphanumeric characters, referred to as the *onion*, printed at the bottom. The onion encodes the permutation of candidate names on the specific ballot relative to a standard *canonical* ordering, encased in multiple layers of encryption using the public keys of the election trustees, as in Votegrity.

The workings of the system are presented in Figure 8.3(b). To vote, Alice marks her candidate choice on the ballot, detaches the two halves and shreds the left side with the candidate ordering. She then scans the right-hand side and takes it home as her receipt. After polls close, all scanned receipts are posted on the bulletin board where Alice can verify her vote is correctly recorded. However, she cannot prove her choice to a third party with just the right half of the ballot since her mark could potentially correspond to any candidate.

Vote processing is analogous to Votegrity. All receipts on the bulletin board are passed through a series of mixes operated by election trustees who use their private keys to successively peel the onion, thereby reconstructing the voter's choice as per the canonical candidate ordering. The mixing process is audited using randomized partial checking. The reconstructed votes are tallied in a straightforward manner.

Verification needs to be performed on another front as well: voters need assurance that ballots are well-formed, i.e., the onions in the lower right half of the ballot correctly encrypt the randomization of candidates on the ballots. Prêt à Voter includes a step allowing Alice to audit her ballot in the polling booth by requesting the system

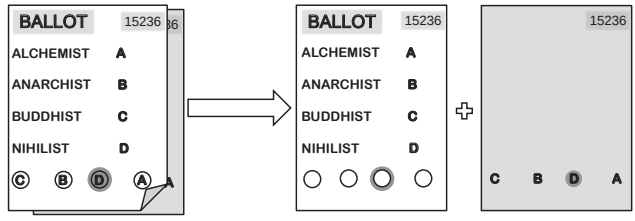


Figure 8.4: The Punchscan ballot (with a vote for *Nihilist*).

to decrypt the onions and prove that they correspond to the candidate ordering. Audited ballots do not count towards the tally and are discarded and Alice may audit as many ballots as she likes until she is satisfied the system is not cheating and then cast her actual vote. The authors also recommend that election authorities publicly audit a set of ballots prior to the election to create public confidence. A sufficiently sized random sampling of ballots should be picked, making it very hard for a malicious party to distribute large numbers of malformed ballots without detection. After polls close, any leftover ballots should be similarly audited for greater confidence.

Prêt à Voter is one of the most prominent E2E voting systems and extensive work has been done in maintaining and extending it. Notable contributions include adapting Prêt à Voter for different electoral systems [299] [581] and voting scenarios [467], enhancing its usability [374] [498] [182] and implementing the system [103].

A variant of Prêt à Voter was used in binding state elections in the Australian state of Victoria in 2014 [125] [183].

8.3.1.3 Punchscan

Punchscan [238], invented by Chaum in 2006, is especially suited for elections where rules mandate a uniform candidate ordering on ballots. Punchscan takes inspiration from Votegrity's notion of layers and from Prêt à Voter's randomization of voter choices on the ballot. The system is described in detail in Chapter 10.

The Punchscan ballot, shown in Figure 8.4, consists of two layers. The top layer has the candidate names ordered using numbers or letters (*a, b, c, d*, etc.) and a set of holes at the bottom. The lower layer contains a random ordering of the same letters. When the layers are overlaid, the letters are visible through the holes. In the polling booth, Alice marks her choice, using a marker to daub the appropriate hole, thereby marking both layers at the same time. The layers are then separated. When considered individually, the top layer only indicates which hole she chose, whereas the lower indicates the letter she selected, and both layers are needed by a viewer to discern her candidate choice. Alice randomly picks one layer to cast and shreds the other. She retains a copy of the cast layer as her receipt.

Vote processing is similar to the systems described earlier. When polls close, Alice can check the bulletin board to verify that her vote was recorded as cast. The ballot serial number, printed in the top right corner on both layers of the ballot, encodes encrypted information which enables election trustees to determine the voter's choice. Punchscan replaces the back-end decryption mixnet of earlier systems with an anonymizing database referred to as the *Punchboard*.

The Punchboard consists of three interlinked tables. The first one lists all markable options on all created ballots, whereas the last lists the candidate corresponding to each option. For instance, for Figure 8.4, the first table will list an entry for the third option which corresponds to a vote for *Nihilist* in the last table. However, the direct links between entries on these these two tables are shuffled via an intermediate table and the shuffling pattern is masked using encryption. This is conceptually similar to how a mix randomizes its inputs but is far more efficient as votes do not have to be encrypted and decrypted multiple times. Instead the system transmits the marks on each receipt over the set of secret paths directly to the appropriate candidates, incrementing their vote count accordingly.

The Punchboard and ballots are generated by election trustees prior to the election and their configuration is kept strictly secret. A preliminary audit is conducted in which half the ballots are randomly picked and publicly revealed along with the corresponding paths in the Punchboard. Any party can verify that votes cast on these ballots would most certainly have been routed to the right candidates and thereby derive confidence in the system. Audited ballots are then destroyed and the remaining half used in the polls. After tallying concludes, the Punchboard is again audited using randomized partial checking, i.e., each path is partially decrypted to prove the system is operating correctly without revealing any end-to-end paths in the process.

Some attacks have been discovered against Punchscan [341] and corresponding procedural safeguards have been recommended (described in greater detail in Chapter 9). Other notable research contributions include adapting Punchscan for remote scenarios [467], reducing opportunities for privacy violation [136], and merging Prêt à Voter and Punchscan for stronger privacy [564].

Punchscan source code was released under an open-source license and the system was trialled in elections of the University of Ottawa's Graduate Students' Association in 2007 [216]. Punchscan was later merged into the Scantegrity voting system.

8.3.1.4 Scantegrity

Scantegrity [146], developed in 2008 by a team of researchers including David Chaum and Ron Rivest, enhances existing real-world voting systems with E2E verifiability. The advantage of this approach is that infrastructure, such as optical scan technology, is already widely deployed and voters are familiar with its usage.

The Scantegrity ballot, depicted in Figure 8.5, lists the candidates with randomly picked code letters assigned to each. A ballot serial number is printed in the top right-hand corner, expressed both in human-readable and barcode format. In the polling

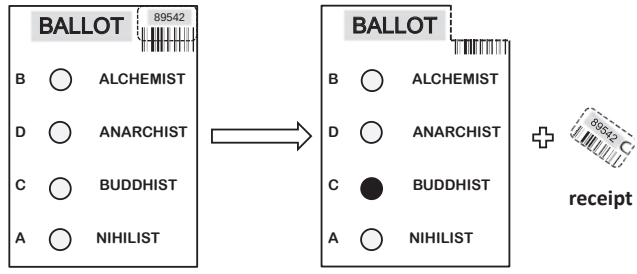


Figure 8.5: The Scantegrity ballot (with a vote for *Buddhist*).

booth, Alice marks the candidate of her choice and scans the ballot into an optical scanner as she would using a regular optical scan voting system. She then detaches the perforated corner of the ballot and notes down on it the code letter for her candidate. This small stub is her Scantegrity receipt.

After polls close, election authorities tally the results of the optical scan system. Scantegrity vote processing runs independently of this count. Election authorities post all ballot serial numbers and the corresponding code letters onto the bulletin board, where voters can verify that their code letters have been correctly recorded by the system. The code letter by itself does not reveal the voter's choice of candidate to a third party as the letters are randomly assigned on ballots. Similar to Punchscan, every code letter on every ballot translates to a vote for a specific candidate in an anonymizing database, in this case referred to as the *switchboard*. Votes are routed accordingly over encrypted paths in a privacy-preserving and auditable manner.

This system suffers a few disadvantages: a coercer may force a voter to mark a certain code letter regardless of which candidate it is assigned to, thereby effectively randomizing her vote. Furthermore, there is no guarantee the voter correctly records the same code on her receipt that she marks on the ballot, and dispute resolution entails a complicated mechanism which involves producing the physical ballot.

Scantegrity II [146] resolves these issues by printing candidate codes on the ballot using invisible ink which are revealed only when the voter marks her candidate choice using a special pen. These codes are randomly assigned and are of sufficient length that they cannot simply be deduced by guesswork. If a voter therefore lodges a complaint citing a valid code, her complaint is very likely genuine. The process is depicted in Figure 8.6 with a vote for *Buddhist*. The voter retains the candidate code *ODX* and ballot serial number *1679-253* as a receipt.

Scantegrity II was trialled in a binding governmental election for the offices of mayor and city council members in Takoma Park, Maryland [526]. The system and the resulting deployment are described in greater detail in Chapter 10. The experience led to further modifications to the system to incorporate features requested by voters and election officials, resulting in Scantegrity III [527].

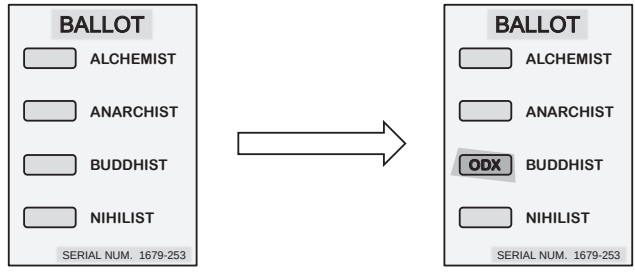


Figure 8.6: The Scantegrity II ballot (with a vote for *Buddhist*).

8.3.1.5 Scratch & Vote

In a paper in 2006, Adida and Rivest [49] noted that mixnets entail complicated mixing and auditing protocols, and require the voter to have faith that election trustees will not collude to subvert the election. To address these shortcomings, they invented Scratch & Vote (S&V), which combines the user-friendly voting experience of Prêt à Voter with the backend simplicity of homomorphic encryption.

Ballots are encrypted using **homomorphic encryption**, a cryptographic primitive which enables them to be tabulated while still in encrypted form. The result of a homomorphic addition on a set of ciphertexts is equivalent to an addition operation performed on the set of plaintexts. Only the result of the addition operation is decrypted, thereby preserving the individual voter's privacy. Homomorphic encryption was first applied to electronic voting in 1985 by Josh Benaloh in his landmark work [161] [91] and is a key ingredient in several voting systems that follow.

To provide voters with further security assurances, S&V distributes the corresponding decryption key among multiple election trustees using threshold cryptography, thereby minimizing risk of abuse of cryptographic credentials. **Threshold cryptography** [457] is conceptually similar to visual cryptography (discussed earlier in Section 8.3.1.1) in that a decryption key is mathematically split into several shares which are distributed among mutually distrusting trustees. The decryption therefore requires the explicit cooperation of a threshold amount of these trustees, and at no time does any party possess the complete key.

Furthermore, vote encryption is also **probabilistic**. In public-key cryptosystems, we recall that the public key used for encryption is not secret, it is accessible to everyone. If Alice's vote were therefore encrypted in a deterministic manner, an attacker could easily mount a **brute-force attack** to deduce it, i.e., he could use the trustees' public keys to encrypt every possible voting option on Alice's ballot and compare the ciphertexts thus generated with those printed on Alice's receipt or on the bulletin board. Probabilistic encryption techniques insert random numbers into the encryption algorithm so that every encryption operation on the same plaintext yields

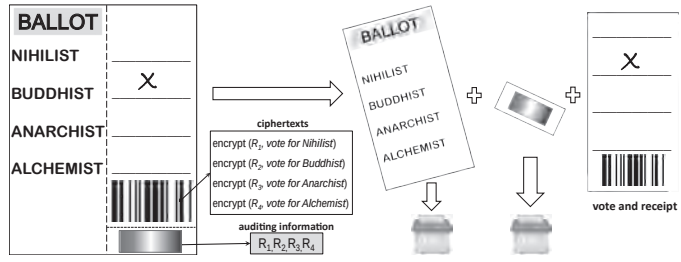


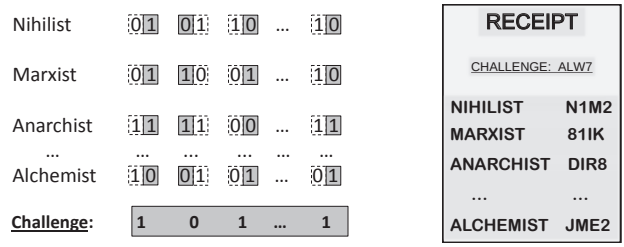
Figure 8.7: The S&V ballot and vote casting process (with a vote for *Buddhist*).

different ciphertexts. These random numbers are large enough that they cannot be simply guessed and they are kept strictly secret. Probabilistic encryption is important in several security protocols and is used in all the voting systems described in this chapter relying on encryption, including the ones described earlier. The reason for highlighting its use here is that, in S&V, the probabilistic nature of the encryption plays a vital role in the vote verification process.

The S&V ballot, depicted in Figure 8.7, detaches into three parts. The left half lists candidate names in randomized order. The right side bears corresponding markable spaces and a barcode. This barcode encodes multiple encrypted data, each ciphertext corresponding to a *vote* for a particular candidate on the ballot. In the lower right corner, a detachable scratch surface conceals the random numbers used to encrypt the barcode data. Any party can therefore scratch off this surface to reveal the random numbers, use them to independently encrypt the candidate choices, and compare it with the barcode data, thereby confirming that the ballot is correctly formed.

At the polling station, every voter is handed two ballots, one to audit and one to cast. Alice randomly selects one, removes the scratch surface and verifies that the candidate ordering corresponds to the barcode data. The authors note that voters generally do not possess the technical expertise to perform this check and suggest that trusted helper organizations may assist them at the polls by providing equipment and guidance. Removing the scratch surface also invalidates the ballot and it can no longer be used. Alice then proceeds to cast her vote with the second ballot, as depicted in Figure 8.7. In a manner similar to Prêt à Voter, she marks her chosen candidate, removes and discards the left side, and hands the ballot to a member of polling staff. He ensures the scratch surface on the ballot is intact, and detaches and destroys it without revealing the underlying content. This step is vital as knowledge of the random numbers would allow a coercer to reconstruct Alice's vote from her receipt. The ballot, now consisting only of the voter mark and the barcode, is cast. Alice is issued a scanned copy as a receipt.

After polls close, Alice can verify on the bulletin board that her vote is correctly recorded. To tabulate results, election staff examine each cast ballot and extract the particular ciphertext from the barcode corresponding to the voter's mark. This ci-



(a) Bit encryption and challenge (b) Receipt

Figure 8.8: Vote verification and receipts for MarkPledge (with a vote for *Anarchist*).

phertext is essentially an encrypted electronic ballot where the underlying plaintext assigns a "1" to the candidate of the voter's choice and "0" for all others. An addition operation is performed over all extracted ciphertexts such that the ones and zeros assigned to candidates in the individual votes are directly summed up. Election trustees then engage in a protocol to decrypt the result using their individual shares of the decryption key, and post cryptographic proofs of correct decryption on the bulletin board. Any observer can easily download the contents of the bulletin board and run his own checks to verify that votes have been correctly summed and decrypted. Individual votes are never decrypted, thereby maintaining voter privacy.

Adida and Rivest also present extensions to this basic scheme, including steps to adapt S&V to a Punchscan ballot, facilitate multiple races and include larger numbers of candidates. They discuss procedural missteps and attacks which weaken the security of S&V and note that these are common to systems like Prêt à Voter and Punchscan, where ballots are filled out by hand.

8.3.2 Precinct-Based Voting with Electronic Ballots

This category consists of systems where electronic votes are cast and tallied in a precinct-based setting. Systems in this category are characterized by inventive combinations of various cryptographic primitives and include influential systems such as MarkPledge and the notion of Voter Initiated Auditing.

8.3.2.1 MarkPledge

MarkPledge [418], invented by Neff in 2004, is, alongside Chaum's Votegrity, one of the earliest E2E voting systems. The key innovation of this system is a novel front-end protocol allowing voters to directly challenge voting machines in the polling booth to prove that they are encrypting votes correctly.

On the MarkPledge ballot, each candidate is assigned a row of *indexes* denoted by pairs of bits, as shown in Figure 8.8(a). A *No* vote is denoted by dissimilar combina-

tions such as (0,1) and (1,0), and a *Yes* vote is denoted by a pairs of similar bits, (1,1) or (0,0). When Alice chooses a candidate, the voting machine will assign her chosen candidate a *Yes* vote and *No* votes to all the rest. The machine then individually encrypts every bit in each index on the ballot and displays on-screen a *commitment* for the chosen candidate. This commitment is essentially a bitstring, a series of bits where each individual bit corresponds to the *Yes/No* content of each index, i.e., *both* ciphertext bits for the chosen candidate. For example, the commitment 1,1,0,...,1 for *Anarchist* (encoded in an alphanumeric format as *DIR8*) implies that the bits, as depicted, are (1,1), (1,1), (0,0),...,(1,1) which are all *Yes* combinations whereas all other candidates are assigned *No* combinations.

As in Scratch & Vote, a probabilistic encryption technique is used to encrypt votes. Once the machine has encrypted her vote, Alice can challenge it by making it reveal the randomness used in encrypting either the right or the left bit of indexes (but not both) for all candidates. This process is depicted in Figure 8.8(a). Alice's challenge consists of a bitstring, where a 0 denotes bits on the left-hand side and 1 denotes bits on the right-hand side. Alice can thus confirm that each revealed bit matches with the commitment displayed earlier by the machine. The machine cannot predict beforehand which bits Alice will choose, and if it has tampered with her vote (by encrypting *No* bit combinations for her candidate of choice), there is a very high probability that the deception will be revealed in the bits that Alice forces the machine to unmask. This mechanism assures Alice that the machine has encrypted her vote correctly and that her vote is cast as intended.

Once the challenge is concluded, the machine then generates *No* votes for all other candidates on the ballot with the important innovation that it populates Alice's chosen right/left bit positions to make it appear that they are *Yes* votes. All this information, including encrypted indexes, the machine's commitment, Alice's challenge, and the random elements are printed on Alice's receipt in the form of short strings of alphanumeric characters, as depicted in Figure 8.8(b). After polls close, all receipts are posted on the bulletin board and voters can confirm that their votes are correctly recorded. Votes are decrypted and tallied using mixnets in an auditable process similar to schemes described earlier in Section 8.3.1.

Alice only needs to ensure that the commitment string the machine originally displayed is printed next to the candidate of her choice (i.e., *DIR8* for *Anarchist* in this case) and she has to retain this string in memory for a short period while challenging the machine. A coercer, examining her receipt at a later time, will only see *Yes* votes for every candidate, and will be unable to deduce her actual vote. Alice alone can differentiate her original *Yes* vote from a fake Yes vote, due to the challenge interaction in the privacy of the polling booth, but is unable to prove it to a third party. This is also the reason that the machine only decrypts the right or left bit position, and not both, during a challenge: if Alice were to record the randomness used to encrypt both bits of an index, she could easily prove her vote to a third party.

MarkPledge is analyzed in greater detail by Adida and Neff [47] who propose that *helper organizations*, consisting of political party members or activist groups,

assist the voter with more detailed verification checks. Joaquim and Ribeiro suggest a mechanism to make the Markpledge cast-as-intended verification process faster and more efficient [325]. This front-end verification technique has also inspired other systems: Joaquim et al. propose two remote E2E voting systems, VeryVote [326] and EVIV (an End-to-end Verifiable Internet voting system) [324] which combine this technique with code voting (described later in Section 8.3.3) and mixnets.

8.3.2.2 Bingo Voting

Bohli, Müller-Quade and Röhrich invented Bingo Voting [108] in 2007, inspired by the concept of a bingo machine as a source of randomness to protect voter privacy.

In the election setup phase, election trustees generate strings of random numbers, such that there is one random number per voter for each candidate on the ballot (i.e., for m voters and n candidates, $m \times n$ random numbers are generated, grouped such that each candidate is assigned one set of n numbers). These numbers, referred to as *dummy votes*, are kept secret, but the trustees publicly commit to them on the bulletin board before elections using a commitment scheme.

A **commitment scheme** is a cryptographic primitive which enables a party to publicly commit to a chosen secret without revealing it until a later time. This may be intuitively visualized with the analogy of a locked box which a *prover* publicly entrusts to a *verifier*. At a later time, the prover uses his secret key to unlock the box and reveal its secret contents. No one else can open the box. Commitment schemes are also *binding* in the sense that the prover cannot alter the contents of the box without being detected by the verifier. In our case, election trustees cannot change the dummy votes once the commitment has been published. Commitment schemes are employed in various security protocols, and applications include timestamping data and trusted computation.

Alice chooses her candidate on a voting machine in the polling booth. The booth also contains a mechanical device, similar to a bingo machine, which Alice uses to generate a fresh random number. The device displays this number and transmits it to the voting machine which assigns it to Alice's candidate choice. The other candidates are assigned numbers chosen from the pool of pre-generated dummy votes. This particular configuration of candidate names and random numbers constitutes Alice's vote and the voting machine issues her a receipt. Alice simply has to confirm that the random number she has generated herself in the booth is assigned to the candidate of her choice on the screen of the voting machine and on her receipt. After polls close, all issued receipts are published on the bulletin board.

Tallying is straightforward. Every time a vote is cast for a certain candidate, all other candidates are assigned dummy votes. In this way, a dummy vote initially allocated to the chosen candidate in the pool of dummy votes is not used. Therefore, the final tally can be computed by examining this pool and the candidate with the largest number of unused dummy votes is the winner of the election.

However, election trustees need to prove that voting machines correctly allocated the dummy votes to candidates. Moreover, this needs to be accomplished without publicly differentiating between dummy votes and voter-generated random numbers on receipts, else voter privacy would be compromised. First, election trustees publish the list of unused dummy votes on the bulletin board alongside their respective commitments and the corresponding reveal information. Next, the trustees publish cryptographic proofs testifying to the integrity of the receipts, i.e., each receipt contains exactly one voter-generated random number, each dummy vote used in the elections was correctly allocated by the machines, and it was used only once. These proofs can be verified by any observer and do not reveal any information which may enable him to identify dummy votes and voter-generated random numbers.

Alice verifies her vote was cast as intended by ensuring the random number she generates in the voting booth is assigned to the candidate of her choice on her receipt. She ensures her vote is recorded correctly by verifying her receipt on the bulletin board. Her privacy is preserved and she is protected from coercion as the newly generated random number assigned to her candidate is indistinguishable from the dummy votes assigned to the other candidates on the receipt. She checks the revealed commitments and the cryptographic proofs to verify that every random number used in the election is strictly accounted for and that the tally has been correctly computed.

Bingo Voting was deployed in student parliament elections in Karlsruhe University, Germany, in 2008 as described in [70]. Another study examines the ramifications of corrupted voting machines on Bingo Voting and prescribes solutions which may be generalized for other voting systems [107]. Liu et al. [369] note that reliance on a random number generator is a potential security vulnerability and devise an alternative solution. Henrich [303] analyzes the feasibility of deploying Bingo Voting for recent real-world elections in the United States, Germany, and India.

8.3.2.3 Voter Initiated Auditing

In most of the systems described thus far, random checks are integral to the security guarantees of the systems. When Alice casts her vote, she has no explicit guarantee that the machine has encrypted her vote correctly. For certain systems like Punchscan or Scantegrity which use physical ballots, she has no direct assurance that the candidate randomization or marking options on her ballot are correctly encoded. Her confidence, instead, derives from the random audits conducted on the system at every key juncture and the knowledge that these audits are overseen by multiple trustees. With these checks, Alice may be certain that any significant malfeasance will be detected.

In contrast to this approach, in 2006, Josh Benaloh introduced the notion of **voter initiated auditing** [84] [85] [86] to give the voter immediate confidence that her vote has been correctly cast. In this paradigm, when Alice chooses her candidate, the voting machine prints an encryption of the vote on her receipt but does not dispense it. Instead, the machine asks Alice whether she wishes to *cast* the vote or *challenge* it. If she trusts the machine, she selects the *cast* option and the printer issues her the

receipt. If Alice is uncertain that the machine has encrypted her vote correctly, she can challenge it to prove it is honest (this step is often referred to as a "Benaloh" challenge). The machine then prints on the receipt, alongside the encrypted vote, the contents of Alice's ballot and the randomness used to generate the encryption. This data allows Alice, or any other party, to replicate the encryption step independently using the election public key, as described earlier in Section 8.3.1.5.

The novelty here is that the machine essentially *commits* to the vote encryption by printing it on the receipt before Alice chooses whether to cast or challenge. If Alice initiates a challenge, she can check if the machine performed the encryption correctly. She is welcome to challenge the machine as many times as she likes until she is confident that the machine is honest before finally casting her vote. If the machine has cheated by encrypting a vote for a different candidate, the odds that it will get caught increase with every challenge.

Voter Initiated Auditing was originally proposed by Benaloh, not as a new system, but as a vote-casting technique to augment existing real-world voting systems with E2E security guarantees, a goal it shared with Scantegrity. It has since then directly inspired other well-known systems such as Helios, VoteBox, and STAR-Vote.

8.3.2.4 VoteBox

VoteBox [510], proposed by Sandler, Derr and Wallach in 2008, attempts to resolve myriad practical issues encountered in building secure and usable voting systems.

VoteBox follows the template set by Voter Initiated Auditing. The system uses homomorphic encryption and the decryption key is predistributed among multiple election trustees using threshold cryptography. On election day, Alice makes her candidate selection on a voting machine, which encrypts her vote and then offers her the option to challenge or cast it. If challenged, the machine prints extra encryption and randomness information on her receipt. Alice takes this to a special terminal in the polling station dedicated to assisting voters to verify challenged votes. Alice may mount as many challenges as she likes before casting her vote.

All receipts are posted on the bulletin board where Alice can verify her vote is correctly recorded. Tallying consists of straightforward homomorphic addition after which election trustees engage in a protocol to decrypt the final result and provide cryptographic proofs attesting to correct decryption. Voters and observers alike may download the contents of the bulletin board and verify the computations.

Apart from its E2E security qualifications, a key contribution of VoteBox is how it brings together various innovations in the research literature to address a range of security and reliability issues regarding the practical side of electronic voting. We list a few highlights: the authors go to considerable lengths to minimize and partition the amount of trusted code running on voting machines. Distinct functions of the voting machine are modularized in hardware and clearly partitioned, thereby enabling administrators to efficiently audit hardware faults. The voting experience is standardized across all voting machines by specifying a graphical user interface

(GUI) consisting of pre-rendered bitmap images which correspond to the actions of voters using the machine [584].

Voting machines log all *machine events* (system messages, the complete record of encrypted votes, and voter challenges and machine responses) using secure logging techniques [511]. All machines broadcast and record event logs over a local network, so that, were a voting machine to fail on election day, all its data up to the point of failure may be retrieved from the storage of other machines. This also makes things considerably more difficult for an attacker who now has to compromise all machines in the precinct to tamper with votes undetected. Furthermore, the local network is connected to the Internet using a data diode [332] which enforces strict one-way data flow, i.e., logs of voting machines in the precinct are publicly broadcast over the Internet, yet remote attackers are not able to hack into the machines.

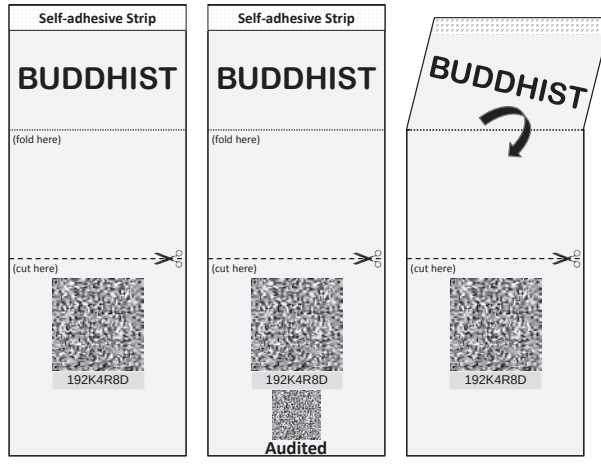
VoteBox has inspired some enhancements: RemoteBox [512] extends VoteBox to secure elections in remote voting facilities such as overseas embassies and consulates. VoteBox Nano [439] reimagines the VoteBox design using only low-cost hardware components, such as field programmable gate arrays. This approach obviates the need for an operating system or software on voting machine, which an attacker may potentially corrupt. Furthermore, any tampering with hardware may be visually detected by polling staff very easily on election day.

8.3.2.5 Wombat

Wombat [80], invented by Ben-Nun et al. in 2011, combines the frontend of Voter Initiated Auditing with Prêt à Voter-style mixnet processing in the backend.

On election day, Alice makes her candidate choice on a voting machine, which prints a *dual* ballot as shown in Figure 8.9(a). Her candidate choice is printed in plaintext followed by a detachable portion bearing her electronic vote, i.e., a barcode representing her encrypted vote. Before dispensing the ballot, the machine offers Alice the choice to challenge or cast her vote. If challenged, the machine prints a barcode with randomness information at the bottom of the ballot, as shown in Figure 8.9(b), which Alice can use to verify her vote. She can mount as many challenges as she likes before casting her vote.

If Alice chooses to cast her ballot, the machine prints "For Casting" on her ballot as in Figure 8.9(c) and dispenses it. Alice folds the ballot as shown, using the adhesive strip to conceal the plaintext. She then exits the voting booth and hands her ballot to a poll worker. He scans the encrypted vote and uploads it to the election website. He next stamps both halves of the ballot and detaches them. The folded plaintext vote is cast in a ballot box, and the barcode is issued to Alice as her receipt. Alice can later verify that her receipt is correctly recorded on the bulletin board. Mixnets are used to shuffle and decrypt the votes which are then tallied. The mixnet protocols also provide proofs of correct operation. All decrypted votes and mixnet proofs are published on the bulletin board where they may be verified by voters and observers.



(a) The ballot (b) Auditing the ballot (c) Folding the ballot

Figure 8.9: Verifying and casting a Wombat ballot (with a vote for *Buddhist*).

Like Scantegrity, Voter Initiated Auditing, and STAR-Vote (described next), Wombat produces both an electronic voting record and a parallel paper trail. This **dual voting** approach has considerable advantages: it is familiar to voters who may be more used to paper-based voting systems. The electronic system may fail, for instance, due to network outages or trustees may misplace cryptographic credentials, in which case the paper votes are a handy backup. Paper trails are also a good tool to audit the elections, and make an attacker's job much harder. To subvert an election, he would need to successfully compromise both the electronic and the paper record.

The team behind Wombat has placed considerable emphasis on practical aspects of the system. Guidelines to implement Wombat securely are detailed in [273]. At an early phase they discovered that giving voters the challenge-or-cast option proves confusing and is unnecessary as only 2-3% of voters need exercise the challenge option to assure election integrity. This inspired the design decision to hide the challenge option, so that voters intending to audit the machine issue the challenge themselves by tapping the screen while the machine is printing the ballot.

Wombat uses Verificatum [577], a free and open-source mixnet solution which has also been used in the 2013 Norwegian Parliamentary elections. The system is supplemented by an open-source Android app for vote verification. Alice can take a picture of the ciphertext and audit information on her ballot using her smartphone and the app verifies that the vote is correctly encrypted and signed. The app also confirms that cast votes are correctly posted on the election website.

Wombat has been deployed in binding elections [80]. In 2011, it was used for student body elections at the Interdisciplinary Center in Israel, involving multiple

races and more than two thousand voters. In 2012, the Israeli political party Merez, used Wombat for intra-party elections, involving approximately 830 voters.

8.3.2.6 STAR-Vote

In 2011, election administrators from Travis County in Texas, seeking to overhaul their election system, reached out directly to the academic research community to assist in building a secure and low-cost voting system. The result is STAR-Vote (STAR stands for Secure, Transparent, Auditable and Reliable) [77]. Like Wombat, STAR-Vote brings together the convenience and usability of electronic voter machines with end-to-end cryptographic verifiability and a paper trail audit mechanism. The system is described in detail in Chapter 14. We provide a brief overview.

At the polling station, Alice is issued a paper slip with her "voting credentials" which identifies the appropriate ballot for her. She inputs the credentials in a voting machine and makes her candidate choice. The machine electronically encrypts and records Alice's vote and issues her a receipt consisting of an encrypted commitment to her vote and related information (such as the ID of the voting machine, the time of voting, etc.). The system also generates a paper trail: the machine additionally prints out a physical copy of the marked ballot. Alice visually confirms her choice is correctly marked and then casts it by hand into a nearby ballot box. The election result is tallied from the electronic vote record. In parallel to E2E verification checks, audits are conducted on the paper ballots to further validate the results.

After casting her vote, Alice may challenge the machine to prove it correctly encrypted the vote. In this case, she informs a poll worker who assists her in verifying that the encrypted commitment on her receipt is indeed correct. Alice may repeat this process until she is satisfied and then cast a final vote. This check is in the spirit of Voter Initiated Auditing but with the key difference that here the challenge and verification steps occur *after* vote casting, not before, and are initiated by voters. This is in the interests of system usability: for voters not interested in challenging the machine, the voting process is fairly similar to that of existing real-world systems.

After polls close, all encrypted votes are posted on the bulletin board where voters can compare against their receipts. Tallying consists of homomorphic addition of the cast ballots. The decryption key is pre-shared among multiple trustees who cooperate to decrypt the result in a verifiable manner. Challenged votes are posted as well, along with verifiable decryptions, and these are excluded from the final tally.

After the elections, **risk-limiting audits** [367] are conducted on the paper ballots. A random selection of ballots is checked on two counts: first, verifying if data on the paper ballots corresponds to the electronic record. The second check determines if the tally of the random sample approximates the electronic results within an acceptable margin. This "belt and suspenders approach" [362] of incorporating multiple safeguards aims at giving voters and observers strong confidence in the system.

The STAR-Vote system architecture inherits several engineering innovations from VoteBox including strict partitioning of program modules, a GUI consisting of pre-rendered graphics, and secure auditable logging of machine events.

STAR-Vote is slated for deployment in elections in Travis County in 2017 [365].

8.3.2.7 DRE-i

Hao, Randell and Clarke noted that voting systems are vulnerable due to their reliance on *tallying authorities*, i.e., trustees tasked with tallying election results who need to be implicitly trusted not to misuse their privileges [290]. Furthermore, as demonstrated in a real-world trial of the Helios system [46], trustees may lack the technical expertise to manage cryptographic credentials securely. To address these concerns, Hao et al. proposed DRE-i (Direct Recording Electronic with Integrity) [288], an E2E voting system which dispenses with tallying authorities entirely. DRE-i is described in detail in Chapter 13.

The DRE-i ballot consists of pairwise strings of encrypted data, or *cryptograms*, assigned to each candidate, depicting *Yes* and *No* votes. When Alice makes her candidate choice on a voting machine, a *Yes* cryptogram is assigned to her candidate and *No* cryptograms to all the others. The machine issues her a receipt listing the candidates with the assigned cryptograms. Individually the *Yes/No* cryptograms do not reveal any information about Alice's choice and resemble strings of random data. Before casting her vote, Alice has the option of challenging the voting machine to prove it is not cheating. If challenged, the machine reveals both *Yes/No* cryptograms for each candidate on the ballot. Any party can now differentiate between *Yes* and *No* votes, and confirm that they are correctly assigned.

All receipts are published onto the bulletin board where they may be verified by voters, alongside proofs that the vote is **well-formed**. This is a privacy-preserving cryptographic proof, used in several of the voting systems described in this chapter, which allows any observer to verify that the ciphertext on the bulletin board contains a vote for only one candidate, without revealing the voter's identity or candidate choice. The votes are then combined in a homomorphic operation to yield the final tally. There is no decryption step. Any observer can therefore download the contents of the bulletin board and replicate the addition to confirm the results.

Self-tallying voting systems have been proposed earlier in the literature, for small-scale scenarios such as boardroom elections [270] [344] [291], but Hao et al. differentiate these from DRE-i which enables self-tallying within the broader framework of E2E verifiable voting, a paradigm they refer to as *self-enforcing electronic voting* [290]. A verifiable classroom voting system based on DRE-i has been implemented and used [287].

8.3.3 Remote Voting with Electronic Ballots

This category comprises systems which enable voters to cast their votes over the Internet and includes prominent systems JCJ/Civitas and Helios. As we noted earlier in Section 8.2.4, remote voting poses unique challenges which restrict the use of some of these systems in politically binding elections.

8.3.3.1 Adder

Adder [343] invented by Kiayias, Korman and Walluck in 2006, is a fully-functional open-source online voting system. Adder is suited to both small and large-scale elections, as well as surveys and data collection applications.

Adder includes a preliminary setup phase in which multiple election trustees engage in a cryptographic protocol to create a public key for the election which is then published on the election website. The corresponding decryption key is distributed among the trustees using threshold cryptography.

To vote, Alice navigates her web browser to the election website, logs in using voting credentials which identify her as an eligible voter, and downloads the election public key. Alice makes her candidate choice and her web browser then generates and encrypts her vote using the election public key and computes and appends a proof of well-formedness. All encrypted votes received by the system are posted on an online bulletin board, where voters can verify they have been correctly recorded.

After polls close, an addition operation is performed over all the encrypted votes on the bulletin board. Election trustees then publicly provide partial decryption information using their individual shares of the decryption key, which enable the server to decrypt the tally. All intermediate computations are posted on the bulletin board.

Adder's security properties are fairly intuitive: the voter encrypts and transmits her own vote and can later verify it is correctly recorded on the bulletin board. Privacy is ensured because individual votes cannot be decrypted unless a threshold amount of these election trustees collude. Voters and observers alike can audit the tallying and decryption process by accessing the bulletin board and verifying each step.

However, Adder is vulnerable to critical security issues common to Internet voting systems. The first is the **untrusted terminal problem**: it is trivially easy for malicious software on Alice's computer to leak knowledge of her vote or switch her vote to another candidate without her knowledge, and she cannot detect this by examining the ciphertext. Second, physical privacy cannot be guaranteed. Someone may be standing at Alice's shoulder while she votes. Furthermore, a coercer, such as a political activist or a family member, may force her to vote a particular way. This threat is also common in other remote voting scenarios, such as postal voting [544].

8.3.3.2 JCJ and Civitas

The system designed by Juels, Catalano and Jakobsson in 2005, commonly referred to as JCJ [335], is notable in that it is the first online voting scheme to offer a high level of coercion-resistance. JCJ combines various cryptographic primitives in a novel protocol that has proved very influential in the research literature.

JCJ assumes that prior to elections, there is an in-person voter registration phase, conducted in a trusted and private environment, where Alice receives a voting credential from the election registrar. This credential is her proof of eligibility and can be used to vote in several different elections. After the registration phase concludes, the registrar publishes an encrypted list of all voting credentials on the bulletin board. Alice then uses a public algorithm to generate any number of *fake* credentials which are indistinguishable from her real one to a third party. She can use these to cast votes under coercion or even freely yield these to coercers to vote on her behalf. Votes cast using fake credentials will show up on the bulletin board as legitimate cast votes but will ultimately be rejected by the system prior to tallying. It is assumed the coercer is not monitoring Alice for the entire duration of the election and she gets some private moments to vote using her real credential.

Votes are cast online. Alice's vote includes her candidate choice and her voting credential and is encrypted with the public key of the election trustees. Also attached is a cryptographic proof enabling any observer to verify that the vote is well-formed and includes a usable credential (real or fake). After polls close, all vote ciphertexts received by the system are published onto the bulletin board where Alice can verify that hers was correctly recorded.

Election trustees then process votes on the bulletin board. Vote proofs are examined and malformed votes or those with unusable credentials are discarded. A **plaintext equivalence test** is used to check if multiple votes have been cast using the same credential. This cryptographic test compares encrypted votes in a pair-wise manner and simply identifies, without decryption, any ciphertexts which include a common credential. No direct information about the credentials or votes themselves is revealed in this test, i.e., voter identities and candidate choices are still secret. If multiple votes using the same credential are identified, they may be processed according to some *revoting* policy such that only one vote per credential is actually counted. For instance, only the most recent vote could be retained and all others using the same credential could be eliminated.

However, this batch still contains votes cast using fake credentials which have to be identified and discarded. All votes are passed through a **re-encryption mixnet**. As opposed to decryption mixes (described earlier in Section 8.3.1.1) which peel off layers of encryption from the vote, re-encryption mixes instead simply inject fresh randomness into the ciphertext. This *re-randomization* generates a batch of completely new ciphertexts which are still encryptions of the original plaintext data, i.e., votes and credentials. These new ciphertexts are then shuffled and output.

The plaintext equivalence test is then again run, this time to compare the encrypted credential in each new ciphertext with the encrypted list of valid voting credentials published by the election authority prior to the election. All votes with fake credentials can then be identified and discarded by the system without revealing the credentials themselves. Leftover votes are then decrypted and tallied in a straightforward manner to yield the election result.

The equivalence tests and the mixing and tallying processes are conducted in a publicly verifiable manner, and all intermediate results are posted on the bulletin board where observers can audit them for correctness. This mix-and-compare process is what gives JCJ the property of coercion-resistance. The re-encryption mixnet effectively breaks the association between cast votes and counted votes such that a coercer has no way of determining if the credential he holds is real or fake and if the vote he cast was included in the final tally or not.

However JCJ has a key disadvantage: the processing time and computation costs of this scheme are very high because compute-intensive plaintext equivalence tests need to be performed for every vote. There have been several suggestions in the literature to optimize this process, including [63], [536], and [275].

Civitas [159], proposed by Clarkson, Chong and Myers, is the first concrete implementation of JCJ along with important modifications, most notably a secure protocol for voter registration. In the original JCJ protocol, a corrupt registrar could simply issue a fake credential to Alice without her being able to detect the deception. However, in Civitas, the registrar functionality is distributed across multiple mutually distrusting trustees. Alice individually contacts these to receive portions of her credential which she then combines into a final credential. The implicit assumption is that some trustees are honest, thereby allowing her to construct the right credential.

Distributing registration functionality for JCJ has also been discussed in [355]. Koenig et al. [350] suggest enhancements, including protection against a potential denial-of-service attack whereby attackers flood the bulletin board with votes with fake credentials, thereby creating severe processing delays. Other proposals include formalizing the voting-process in Civitas, a user interface inspired in part by Helios (described next), and credential management using smartcards [411] [408].

The core JCJ architecture has inspired other E2E voting systems which improve upon its various usability aspects. For instance, Selections [158] provides an easier way for voters to register and manage credentials by incorporating the use of *panic passwords*, i.e., voting credentials that the voter may create on the fly. Trivitas [123] adapts Civitas to further simplify the non-intuitive and complex verifiability process for the layman voter by introducing the notion of *trial votes*. These are cast alongside regular votes, and are processed in much the same way, except that trial votes are decrypted and revealed at different stages, enabling the voter to derive strong confidence that her actual vote is being correctly handled. Caveat Coercitor [269] extends JCJ's coercion-resistance property for more general scenarios including leakage of voting credentials by malware, corrupt registrars and impersonation attacks.

8.3.3.3 Helios

Helios [44], invented by Adida in 2008, is an online voting system intended for scenarios where privacy and integrity concerns are paramount but the risk of coercion is low, i.e., in settings such as student body elections, clubs and online communities. Helios does not introduce any novel features as such, but successfully brings together different innovations in the literature to build an efficient and highly usable voting system. Helios is described in detail in Chapter 11. We present a brief overview:

To vote, Alice uses her web browser to navigate to the election website and marks her choice on an online ballot form. Her vote is encrypted locally on her computer, a digital receipt is printed on the screen. In the spirit of Voter Initiated Auditing, Alice is then offered the opportunity to cast or challenge her vote. She can challenge the system multiple times until she is convinced it is encrypting her choice correctly. When she is ready to cast her vote, she authenticates herself to the system as an eligible voter and uploads the ciphertext. The system posts it on an online bulletin board where she can verify it at a later time against the receipt she received earlier.

After polls close, encrypted votes on the bulletin board are processed using re-encryption mixes to effectively break the association between voter identity and cast votes after which votes are decrypted and tallied. The initial version of Helios employed the Sako–Kilian mixnet protocol [508] which published cryptographic proofs allowing third parties to confirm that the mixnet processed the data correctly. In 2009, Helios was updated to Helios 2.0 which improved efficiency and voter privacy by replacing the mixnets with a homomorphic encryption scheme and distributing the election decryption key among multiple election trustees.

Helios provides E2E guarantees but, as noted earlier, is not suited for political elections where the risk of coercion is high. Adida acknowledges this as a general limitation of remote voting systems and suggests that Helios might in fact educate voters on this threat. For this reason, the initial version of Helios included a "Coerce Me!" button, which Alice could press when casting her ballot, thereby emailing a coercer complete details of how she voted. Adida also notes that a corrupted version of Helios may impersonate absent voters and cast ballots on their behalf, making it all the more important that voters engage in the verification process.

Helios has proved very influential and is actively being maintained. The source code is available under an open-source license. Helios has been the subject of multiple usability studies [572] [338] [40]. A number of vulnerabilities have been discovered in the implementation of Helios [222] [302] [99]. Examples include cases where attackers may copy and re-cast previously cast votes [173], and a corrupted system may dispense identical receipts to different voters while modifying their actual votes undetected [357]. Many of these issues have been fixed in later versions of Helios. Improvements have also been suggested: Helios has been modified to enable new properties such as self-tallying [207] and stronger privacy guarantees [97] [196].

Helios has been used in several binding elections: the Université Catholique de Louvain used Helios in 2009 to elect the University President in an election with

Candidate	Vote Code	Acknowledgment Code
ALCHEMIST	5962	218931
ANARCHIST	2168	854269
BUDDHIST	3756	129853
MARXIST	1247	875391
NIHILIST	9881	039852

ID: 4896327

Figure 8.10: Sample code sheet.

about 5000 registered voters [46]. Helios has also been deployed for student elections at Princeton [399] and internal elections of the International Association for Cryptologic Research (IACR) [26] and the Association for Computing Machinery (ACM) [42]. Helios has also been adapted to accommodate different voting schemes such as single transferable vote (STV) and ranked voting schemes. Examples include Zeus, built by Louridas et al. [371], and another variant developed by Bulens et al. [121]. Both of these systems have been deployed in dozens of university-level elections.

8.3.3.4 Pretty Good Democracy

Pretty Good Democracy [506] is an online voting system developed by Ryan and Teague in 2013. Pretty Good Democracy relies on threshold cryptography and code voting to provide a subset of E2E verifiability properties.

Code voting was first suggested by Chaum as a solution to the untrusted terminal problem [142] and now serves as a key component in several online E2E voting systems [326] [358] [586] [324] [459]. The key idea is simple: codes are used to set up a private authenticated channel between voters and election trustees. Prior to polls, the trustees send each voter a physical *code sheet* in the mail which assigns unique, randomly picked *vote codes* and *acknowledgment codes* to each candidate on the ballot as depicted in Figure 8.10. These codes are alphanumeric strings of a length that resists simple guessing attacks. Each code sheet bears a unique ballot ID.

To vote, Alice logs onto the system, and inputs the ID on her code sheet, followed by the vote code for the candidate of her choice. For example, she would type in *2168* to vote for *Anarchist*. The system responds with the corresponding acknowledgment code, in this case, *854269*. Since the codes are randomly assigned and known only to Alice and the trustees, a malware on her computer or an attacker eavesdropping on network traffic cannot deduce her vote nor alter it in any meaningful way without being detected. Furthermore, Alice's vote code assures the trustees that they are communicating with Alice herself, since it should be difficult for an attacker to procure her code sheet. Likewise, if Alice receives the correct acknowledgment code,

she can be certain her vote was correctly received by the trustees and not an attacker masquerading as a trustee. If either party enters a wrong code, the protocol fails.

After polls close, ballot IDs for all votes received by the system are published on the bulletin board and Alice can check that hers is included in the list. Decryption mixnets are then used to anonymize votes prior to tallying, similar to Votegrity and Prêt à Voter, and the process is likewise audited using randomized partial checking.

The bulletin board does not display vote receipts otherwise it would be trivially easy for Alice to prove her vote to a third party using her code sheet. However, displaying only ballot IDs of cast votes opens up the possibility that a corrupt tallying authority may alter votes undetected. To address this vulnerability, Ryan and Teague employ threshold cryptography to distribute system functionality among the election trustees. When the system receives Alice's vote code, it engages in a cooperative protocol with a threshold number of trustees to generate the appropriate acknowledgment code and record Alice's vote in the system. It is therefore not possible to alter votes unless a large number of these trustees are corrupt, which should reassure Alice to a degree that her vote has been correctly recorded by the system.

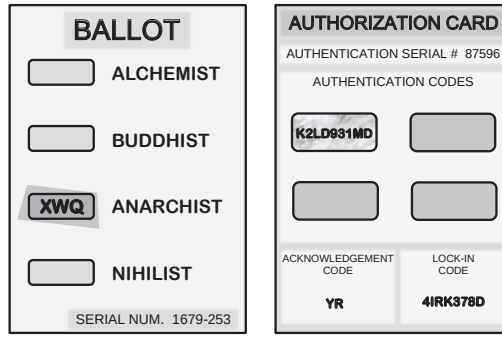
However, Pretty Good Democracy only partially addresses the problems associated with remote voting. There is still the threat that someone may be standing at Alice's shoulder when she votes and may even coerce her to vote for a certain candidate. Alice may sell her vote by giving her code sheet to someone else. An online attacker may pose as a fake election server and her vote may be lost. Similar attacks have been observed in remote voting scenarios such as postal voting. The authors, therefore, explicitly recommend that Pretty Good Democracy only be used for low-risk elections such as for student bodies and professional societies and not for politically binding elections.

Pretty Good Democracy has inspired other systems: Pretty Understandable Democracy [120] simplifies the way votes are processed. Virtually Perfect Democracy [78] introduces personalized voter smartcards into the protocol to increase trust in the system and provide an element of coercion resistance. Pretty Good Democracy has also been adapted for ranking-based voting schemes such as single transferable vote (STV) and instant-runoff voting (IRV) [301].

8.3.3.5 Remotegrity

Remotegrity [586] is an online E2E voting protocol, proposed by Zagorski et al. in 2013. Remotegrity is not a full-fledged system but an extension to precinct-based systems like Scantegrity and Prêt à Voter, specifically aimed at absentee voters, and uses an innovative combination of code voting and scratch surfaces to ensure votes are securely recorded.

As an absentee voter, Alice receives via post a Scantegrity ballot (described in Section 8.3.1.4) and a Remotegrity authorization card, as depicted in Figure 8.11. The authorization card contains an authentication serial number, and three types of codes: multiple authentication codes, a lock-in code, and an acknowledgment code.



(a) Scantegrity II ballot (b) Remotegrity authorization card

Figure 8.11: Voting with Remotegrity (casting a vote for *Anarchist*).

To vote, Alice logs on to the system and enters her ballot serial number and the authentication serial number on her authorization card. This allows the system to check that she has not already cast a vote. Alice next obtains the vote code on the ballot for the candidate of her choice (*XWQ* for *Anarchist*). From the authorization card, she scratches off an authentication code at random. These two codes are sent to the system as a single message and are posted on the bulletin board. The election trustees verify that the authentication code is valid. They then append an acknowledgment code to it, digitally sign the whole message, thereby certifying it as genuine, and update the entry on the bulletin board. At a later time, Alice checks the bulletin board and verifies that her vote has not been maliciously altered and confirms the acknowledgment code and the trustees' signature. She then approves the updated entry by scratching off and submitting the lock-in code on her authorization card. The system again updates the bulletin board entry to include the lock-in code.

When polls close, the trustees check all entries to verify the lock-in codes are correct. They then input the ballot serial numbers and the corresponding vote codes for each entry into the Scantegrity tallying system. These votes are then tallied alongside precinct-cast votes using the Scantegrity Switchboard in a publicly verifiable manner (as described in Section 8.3.1.4 and Chapter 10).

The use of codes effectively defeats malware and hackers who may tamper with Alice's computer. The scratch surfaces act as tamper-evident seals. If the election trustees were to cast a vote on Alice's behalf, Alice could produce her authorization card with the authentication code scratch surfaces still intact as proof of malfeasance. If they were to change her vote after she submitted it, she could refuse to commit to it with her lock-in code. If the trustees were to append the lock-in code themselves, her authorization card, with the lock-in scratch surface still intact, would prove that she did not authorize that vote. The trustees may issue authorization cards with invalid codes, but this may be detected by conducting public audits in which randomly chosen authorization cards are examined for integrity. The protocol therefore en-

(a) The ThreeBallot ballot

(b) The VAV ballot

Figure 8.12: Voting with ThreeBallot and VAV (with votes for *Nihilist*).

sures that Alice's vote is cast and recorded by the system as she intended. Regarding the verifiability of the tally, Remotegrity inherits Scantegrity's security guarantees. However, Remotegrity does not protect against vote coercion or vote buying.

Remotegrity was successfully trialled in municipal elections in Takoma Park, Maryland, in 2008 [586]. The system is described in detail in Chapter 10.

8.4 Non-Cryptographic E2E Voting Systems

These systems represent an interesting experiment in the overall research on E2E voting. Systems such as ThreeBallot, Twin and Aperio emulate E2E verifiability properties in a wholly precinct-based paper-ballot format without resorting to any cryptography. This approach has only produced a handful of workable systems thus far, but it has generated valuable new insights about E2E voting in general. Additionally, due to their exclusion of complex cryptography, these systems aid in communicating fundamental principles of E2E verifiability to the layman.

8.4.1 ThreeBallot, VAV and Twin

ThreeBallot [487], introduced by Rivest and Smith in 2007, derives its name from its unique ballot design, consisting of three detachable ballots, as depicted in Figure 8.12(a). Each ballot is assigned a unique serial number printed at the bottom. In the polling booth, Alice marks the ballots such that each candidate is marked once whereas the candidate of her choice is marked *twice* Figure 8.12(a) represents a vote for *Nihilist*. Alice's vote can therefore be understood as a composite of the three ballots. The ballots are then detached. Alice randomly chooses one as a receipt and scans a copy to take home. She then casts all three ballots into the ballot box.

After polls close, all cast ballots are published on the bulletin board, and Alice uses her receipt to verify the corresponding ballot is correctly recorded. The tally is computed by simply adding up all ballots on the bulletin board. Since each candidate receives one vote by default, the result is inflated by the number of voters, which is accordingly subtracted from the tally. Any observer can replicate and verify the tally.

An important distinction to be made here is that Alice's receipt is not an encryption of her vote. Instead, the receipt testifies to a key component of the vote which has been delinked from the other components. Alice's privacy is maintained as the receipt reveals nothing about her actual vote unless examined in conjunction with the two other ballots she cast, and these could be any on the bulletin board. Furthermore, if any party tampers with her ballots, Alice has a one in three chance of detecting it on the bulletin board. For large-scale tampering, the odds are much higher.

ThreeBallot's non-intuitive ballot marking scheme may impact usability: in a mock election, a large number of voters experienced difficulty using ThreeBallot and more than 30% of initial cast votes proved invalid [333]. There are also security issues: a coercer may force Alice to mark her ballots in certain unique patterns which he can later identify on the bulletin board. These issues and potential solutions are documented in [487] [59] [304] [155] [341] and are also discussed in Chapter 9.

Rivest and Smith also presented two systems that are variations on ThreeBallot, VAV (Vote/AntiVote/Vote) and Twin [333]. The VAV ballot suite, depicted in Figure 8.12(b), consists of two *Vote* ballots, and one *AntiVote* ballot. The marking process differs from ThreeBallot in that only one candidate is marked on each ballot. The *AntiVote* ballot essentially *cancels* out one *Vote* ballot, and both have to be marked identically. The leftover *Vote* determines Alice's actual choice. Figure 8.12(b) depicts a vote for *Nihilist*. Alice scans one of the three ballots at random to use as a receipt. Tallying and verification processes for VAV are very similar to ThreeBallot.

Twin is based on traditional voting systems in that the voter casts only one ballot, but with the innovation that Alice does not take home her own receipt but that of another voter. In this scenario, a big bin in the polling station is periodically filled with valid receipts. After casting her vote, Alice makes her way to the bin, draws out a receipt at random, and takes a copy home, which she later verifies on the bulletin board. These receipts are therefore referred to as *floating receipts*. This system is very easy to use, and though individual vote verifiability is sacrificed, voters can still verify the election tally and coercion resistance is ensured.

8.4.2 Randell & Ryan's Scratch Card Voting System

In 2006, Randell and Ryan proposed a voting system [478] that augments the Prêt à Voter ballot with scratch surfaces and dispenses with cryptography and mixnets.

The ballot, depicted in Figure 8.13, is detachable in two halves along the middle. Candidate names are listed in randomized order on the left side with corresponding markable spaces on the right. The Vote Identification Number (VIN), functioning as

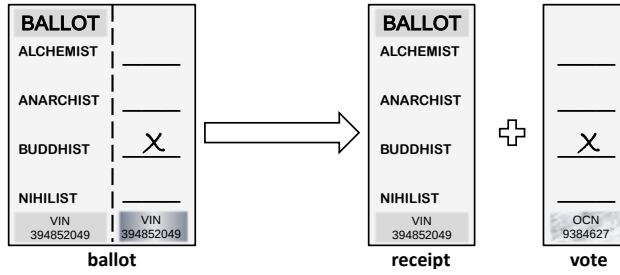


Figure 8.13: The Randell & Ryan Scratch Card Voting ballot.

a unique serial number for the ballot, is duplicated on both halves, except on the right it is printed atop a scratch surface. The scratch surface conceals an encoding of the candidate ordering on the ballot, referred to as the Order of Candidate Names (OCN).

On election day, Alice marks her candidate choice on the ballot, detaches the right side and casts it in the ballot box. The left side serves as her receipt. Election staff will accept her vote only if the scratch surface is still intact. After polls close, VINs of all cast votes are published on a bulletin board and Alice can ensure the system received her ballot. To process votes, election officials first remove the scratch surface of each cast vote. This reveals the OCNs, enabling them to reconstruct the original candidate ordering on the ballots, and thereby deduce the voters' choices and compute the tally. More importantly, removing the scratch surface also destroys the VIN on top, protecting voter privacy. Once the VIN is removed, Alice's vote can no longer be differentiated from other cast votes.

The OCNs are simple codes, each of which maps to a different candidate ordering. Prior to casting her vote, Alice can audit her ballot by simply removing the scratch surface, decoding the OCN and verifying if it matches the candidate ordering on the ballot. However, removing the scratch surface invalidates the ballot, and she will need a new one to cast her vote. Alice can audit as many ballots as she likes until she is convinced the ballots are correctly formed, and then cast her vote.

This system provides a subset of E2E properties. Only received VINs are published on the bulletin board, providing Alice assurance that her vote has been received, but not that it has been recorded and tallied exactly as she cast it. Cast ballots are not published as it would make it trivially easy for any party to deduce voters' choices from their receipts. Furthermore, removing the scratch surface irretrievably destroys the link between cast votes and counted votes, and Alice has no means of verifying that election officials correctly included her vote in the tally and did not replace it with a fake one. To increase transparency, Randell and Ryan suggest instituting random audits, and incorporating paper trails into the system.

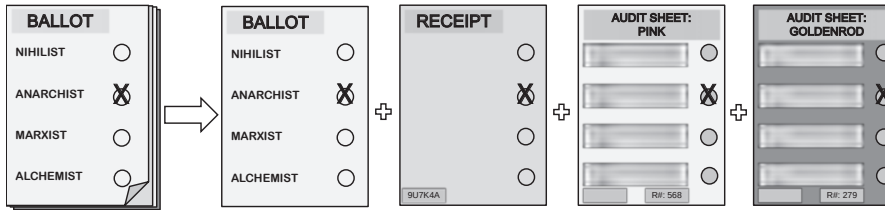


Figure 8.14: The Aperio ballot assembly with a vote for *Anarchist*.

8.4.3 Aperio

Aperio[215], developed by Essex Clark and Adams in 2010, is intended for low-tech minimal environments, such as elections in developing countries. Aperio derives inspiration from Punchscan in its use of ballots with stacked sheets which enable audit trails to defend against ballot box stuffing and ballot tampering.

The Aperio ballot suite, depicted in Figure 8.14, consists of a ballot with randomized candidate names, a receipt layer and multiple distinctly colored audit layers (we assume two audit layers in this example, colored pink and goldenrod). The sheets are backed with carbon paper so that marks made on the ballot are copied onto the lower layers. Receipt and audit sheets each bear a unique reference number, the receipt serial number and the audit reference number, respectively.

In the election setup phase, election trustees assemble the ballot suites by superimposing the individual sheets and binding them together. They also generate two lists for each individual audit layer: a *receipt commitment list* linking each receipt serial number to an audit reference number, and a *ballot commitment list* linking the audit reference number to ballot candidate ordering. Figure 8.15 includes sample entries for these lists for the ballot suite in Figure 8.14. These commitment lists are secured in separate tamper-evident envelopes and placed in safe custody prior to polls.

To cast her vote, Alice marks her candidate choice on the ballot suite and hands it to a member of the polling staff who ensures that the counterfoil is still intact. If not, the vote is rejected. If intact, the staff member then separates the four sheets. The ballot is cast in the ballot box, the receipt is issued to Alice to take home, and the audit sheets are cast in corresponding pink and goldenrod-colored audit boxes. After polls close, results are tallied by simply counting the votes in the ballot box.

To audit the election, the trustees retrieve the receipt and ballot commitment lists. They check the envelope to ensure there has been no tampering. A coin is publicly tossed, and, depending on the result, one of the two colored audit boxes is used to generate a *receipt audit trail* and the other to generate a *ballot audit trail*. For example, if the pink box is chosen to audit receipts, the envelopes containing the pink receipt commitment list and the goldenrod ballot commitment list are opened.

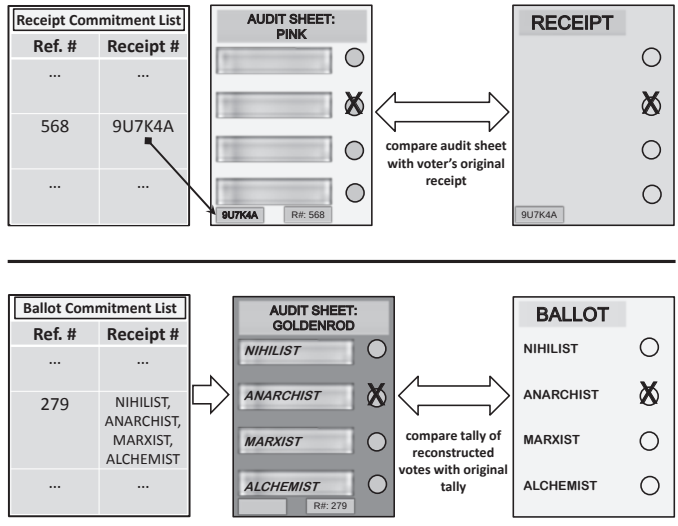


Figure 8.15: Auditing receipts and ballots in Aperio.

The envelopes containing the pink ballot commitment list and the goldenrod receipt commitment list are destroyed.

The commitment lists and audit sheets enable individual reconstruction of the vote and the receipt, as depicted in Figure 8.15. The pink audit sheet already bears the mark for Alice's candidate choice. Election staff search for the sheet's audit reference number (*568* in our example) in the receipt commitment list and note the corresponding receipt serial number (*9U7K4A*) by hand on the sheet. This step effectively reconstructs Alice's receipt. Reconstructed receipts are posted on the bulletin board where voters can verify them against the receipts issued to them earlier.

Likewise, the reference number on the goldenrod audit sheet (*279* in this case) identifies a specific candidate ordering in the ballot commitment list. Trustees write this ordering on the sheet, thereby reconstructing the vote. These votes are then counted independently and results are compared with the official tally.

Aperio's key novelty is that receipts and ballots are audited separately. The act of destroying the unused commitment lists effectively decouples the ballots from the receipts, thereby preserving voter privacy. Alice can verify her vote was correctly recorded by comparing her receipt against the reconstructed receipts. Election trustees and observers ensure the votes have been correctly counted if the tally of the reconstructed votes matches the original results. However, as in the case of paper-based E2E systems, certain coercion attacks, such as randomized voting, still apply.

Aperio has inspired an electronic counterpart, Eperio [218], which employs cryptographic primitives to similarly verify votes against commitment lists. Another sys-

tem, Hash-Only Verification (*Hover*) [219], uses this verification approach in conjunction with the Scantegrity optical-scan system to reduce reliance on election trustees and make the voting system intelligible to non-technical voters.

8.5 The Way Forward for E2E Voting Systems

As described thus far, considerable work has been done on developing E2E voting systems and reconciling theoretical notions with practical realities. However, to fully transition these systems into the real world, a host of pressing technical issues, usability concerns and legal dilemmas still need to be resolved. In this section, we briefly overview progress and outstanding challenges in these domains.

8.5.1 Technical Issues

As often happens when prototyping theoretical systems, the implementation itself may introduce security flaws. Karlof et al. [340] analyzed two systems, Votegrity and MarkPledge, and described several scenarios where voter privacy and election integrity may be compromised by exploiting flaws in how the systems were built. For instance, malicious software on voting machines could encrypt votes in a manner that surreptitiously leaks information regarding the voter's candidate choice. Malicious voting machines may sabotage an election by issuing large numbers of forged or invalid receipts. Large-scale attacks would be detected but may negatively impact public confidence in the system. Human factors also need to be considered. For instance, voters who do not intend to verify their votes may discard their receipts on polling premises where a poll worker could collect them and later manipulate the corresponding votes without fear of detection. Various such attacks and proposed solutions are described in Chapters 7 and 9.

Researchers have presented various practical guidelines to implementing voting systems: Karlof et al. advocate a **systems approach** [340] which essentially states that security analyses of such systems should not examine system components in isolation but consider the entire system and how the individual parts interact with each other. Bruck et al. strongly caution against "all-at-once" approaches [117] to building voting equipment where diverse functionality is bundled together into the same hardware components. They propose a highly **modular architecture**, dubbed *Frog voting*, which clearly partitions processes such as voter registration, vote recording, casting, tallying, and auditing, thereby reducing complexity, simplifying certification, and encouraging incremental innovation. Rivest and Wack have proposed the principle of **software independence** [486] (detailed in Chapter 1) urging that the integrity of election results should not be dependent on the software used by the system. Popoveniuc et al. [466] also detail a series of security checks which ensure E2E verifiability properties of a voting system.

A second major direction of research has been to improve individual components of voting systems. There is a significant body of work on **securing electronic voting machines**. For instance, researchers have proposed the use of trusted hardware, i.e., specialized tamper-proof chips which protect cryptographic credentials and guarantee that voting machines run trusted software [456] [237] [532]. Researchers have also identified critical security properties that voting machines must ensure, and proposed corresponding solutions [396] [101]. Another key component in E2E voting systems is the **public bulletin board** and various solutions have been put forward to implement it [300] [185] and make it robust against attacks [350] [274].

Online voting has also received considerable attention. Proposed methods to defeat hackers and malware include the use of visual cryptography techniques [455], CAPTCHAs [468], voice recognition [464] and deploying assistive handheld devices [582]. However, voter privacy and coercion concerns still persist.

Unfortunately, most of these solutions have been proposed piecemeal and it is unclear if and how they will fit together. There have been very few attempts, like VoteBox (described earlier in Section 8.3.2.4), to harmonize these various innovations in the context of a complete E2E voting system. Substantial work is still needed in this domain before E2E voting systems are deployed for mainstream use.

8.5.2 Usability

The importance of usability for a voting system cannot be emphasized enough and usability oftentimes conflicts with security properties. Ensuring system usability is also complicated by the fact that elections occur only rarely and voters must be able to vote with near 100% success while having little or no experience or training on that voting system. This problem also extends to poll workers and election officials who may have limited technical expertise and experience regarding certain systems. For these reasons, the National Institute of Standards and Technology (NIST) has advocated a broad all-encompassing perspective to study usability within the context of the complete voting system, including the physical environment, the voting product, the ballot, the voter and all personnel involved in the process [360].

At this particular stage in the development of E2E voting systems, the central usability concerns are whether voters can successfully cast their votes using these systems, and more importantly, whether or not they are able to undertake the verification process. Regarding vote casting, evidence suggests that certain design features of these systems are problematic. In one study, the ballot design of Prêt à Voter, requiring voters to shred half of their ballot to ensure vote privacy, caused confusion [516]. Another study involving Helios, Prêt à Voter and Scantegrity II found that a significant number of voters failed to cast a vote with each system [40]. Troublingly,

many of those voters thought they had in fact successfully cast a vote. It also took almost twice as long to cast a ballot as with a traditional paper-based system.[5]

Individual vote verification is a new and novel concept for voters and adds a layer of complexity. Critically, voters are active participants in auditing the election and certifying its results, and if, for whatever reason they are unable to verify their votes, the system's security guarantees become moot and the system is not auditable. Recent testing and "live" applications of E2E systems have resulted not just in consistently low rates of voter verification but even lower rates for those who actually report discrepancies [195] [395]. This is likely due to two reasons.

First, voters do not see the need to verify their vote. One possible explanation for this is that **trust-transference** from election authorities to voter does not happen. As Olembo et al. [440] discovered, voters may initially verify their vote out of curiosity, but after continued use of the system, they establish trust in the system and verification is deemed unnecessary. In line with this conclusion, recent usability testing involving Helios and a Galois-designed prototype based on STAR-Vote, found that participants had a tacit level of trust in any voting system provided it was officially branded, e.g., *Jurisdiction X Official Election Website* [402]. The participants considered verification unnecessary if a voting system met this basic criterion.

The second concern is complexity. The verification process of Helios has repeatedly been found to be difficult [339] [402]. The reasons may be technical or even that the voters may not understand what to do since the verification language is not related to prior voting experience on their part. Research suggests that terms like "audit," "verifiability" or "ballot fingerprint" lack clarity for voters, cause confusion and do not engender trust in the voting system [441]. This is critical as voters need to understand the necessity of verification to be motivated to invest the extra effort in verifying their vote. The intuition behind verification also needs to be effectively communicated. As Schneider et al. discovered in their study concerning Prêt à Voter, simply confirming that an encrypted vote on a bulletin board corresponded to a receipt did not provide sufficient security guarantees for voters themselves [516].

Several technical solutions have been proposed in the literature, including simplifying security elements or enhancing voting systems to facilitate individual vote verification [404] [498], bundling multiple receipts to enable mass verification [107], or involving third parties in the process, such as activist groups or helper organizations [49] [47], but these require further study from a usability perspective.

Overall, research in the usability of E2E voting systems is still in an early stage and significant work is needed to demonstrate that these systems are accessible to non-expert users while simultaneously maintaining the desired security properties.

[5]However, we would note here that the Scantegrity team has critiqued the implementation of the Scantegrity system used in this study [386].

8.5.3 Legal Framework

It is a common perception that numerous attempts to introduce new voting technology, for example in Britain, Finland, Ireland and Quebec, have failed in large part due to the absence of a comprehensive supporting legal framework [55]. Indeed, many of the issues regarding new voting technologies are not due to the technology itself, but rather the lack of ancillary processes that establish trust in the voting system. A new paradigm, like E2E voting, requires legislators to craft an appropriate legal framework that embeds election norms and details administrative procedures to address risks, problems and threats. This includes aspects such as testing requirements, implementation instructions and processes for problem resolution.

A useful guiding principle in constructing a legal framework is that of **functional equivalence**. Any new legal framework must essentially function in an equivalent way to the legal framework for existing voting systems, i.e., the new framework must be at least as good, if not better, than the old one. Towards this end, legislators need to identify the distinct purposes served by existing voting systems and ensure that new rules also address the same needs. Applying this criterion to E2E voting systems requires consideration that current electoral values and community expectations are preserved, while making specific legislative changes to ensure that voting experiences using E2E voting systems are as good as or better than existing procedures.

Another overarching concept is that of **nondiscrimination** [370]. A new legal framework should not have the unintended consequence of discriminating against any voter. For instance, in the United States, any electronic technology, including voting equipment, needs to be Section 508 compliant [433], i.e., it must attempt to fully accommodate disability groups. If E2E voting systems cannot successfully cater to certain voter groups, legislation must set out clear alternatives to avoid discriminatory legal challenges. A good example is the case of absentee voting, where eligibility requirements are defined, permitting certain groups to cast their votes remotely. Furthermore, legislation should not unduly limit technological choices to certain vendors or technology in cases where a better technology or solution exists. Excessive detail in legislation can inhibit innovation and create legal *technology locks* [55].

An additional function of a legal framework is to **minimize and mediate error and risk**. No electoral mechanism, be it electronic or paper, is ever absolutely secure from every possible error or risk, and the greater question is whether the regulatory framework provides voters and political stakeholders confidence that risk is minimized and contingencies are in place to cover the range of possible circumstances.

Regarding specific properties to be included in legislation, Schwartz and Grice detail the following **normative values** addressing electronic voting systems [520]:

- Accessibility and facilitate reasonable accommodation
- Voter anonymity
- Fairness
- Accurate and prompt results

- Comprehensible and transparent processes
- System security and risk assessment
- Detection of problems and remedial contingencies
- Legislative certainty and finality
- Effective and independent oversight
- Cost justification and efficiency

Many of these values could be embedded into a legal framework in the form of technological choices within the specifications of an E2E voting system. For instance, to deter coercion, voters may be permitted to change or update their vote once it has been cast, as is the case of elections in Estonia [376]. Similarly, if multiple modes of voting are available, legislation may specify that an electronic vote can be revoked if the voter casts a paper ballot. Other technological specifications that may be considered election norms include the capability to detect undervotes and overvotes on marked ballots prior to casting and the ability to cast a blank ballot as a protest vote without it being considered technically invalid.

It should be recognized that constructing a legal framework involves trade-offs between competing norms. An illustrative example is the case of Dutch voters and the Rijnland Internet Election System (RIES) [246], an online voting system used for the Rijnland District Water Board Elections in 2004. In the interest of transparency, most of the RIES technology was made open-source. RIES provides a degree of verifiability but weak privacy. Upon voting, the system generated a technical code, enabling voters to verify their vote was included in the tally. However, if a voter disclosed her code, any third party could determine her actual vote. In terms of norms, therefore, transparency trumped voter privacy. Whereas this particular example may not directly apply to most E2E voting systems, where cryptography obfuscates the contents of the voter receipt, it is indicative of the conflicts legislators will have to wrestle with in preparing a legal framework for these new systems.

8.5.4 Uptake of E2E Voting Systems

Broadly speaking, uptake of a new voting system depends on how successfully it incorporates existing election norms, and equally importantly, how effectively this perception is communicated to voters. Research on Internet voting lists four factors which may serve as a good starting point to discuss the uptake of E2E voting systems [445]. These are security, privacy, accountability and economic feasibility.

Regarding security and privacy, in Section 8.2 we have already motivated the fundamental requirement for voting systems to ensure vote privacy and election integrity. However, security also needs to be considered in terms of **reliability**, i.e., how a system isolates and reacts to failure. While a comprehensive legal framework will considerably ameliorate failure reaction, a voting system should be designed so that there is no single point of failure. As a risk assessment of large-scale events notes [311]: “If failure in one part of an information system can cause failure in other in-

terconnected parts, then the system is susceptible to cascade failure." Furthermore, in the words of Pieters and Becker [461]: "It is not only important that a system is reliable, it is also important that people believe that the system is reliable."

Fostering a common understanding of the risks associated with a voting system may actually advance acceptance. Risk exists even in paper-based elections but these are generally known, accepted and tolerated risks. Transparency and accountability are key to minimizing risk, and risk perception, by detailing and perhaps even publicly demonstrating potential risks and corresponding remedies. The legal framework should require the electoral authority to establish a set of procedures and tests to be completed before the voting system goes live. Additionally, procedures should be legislated that provide solutions for potential risks and these should be updated before each election to ensure protection against new threats or vulnerabilities.

Moreover, comprehensive and high-quality voter instruction is critical to uptake of a new system and typically falls under the auspices of the legal framework ensuring equal access. Voter education is particularly important for a new paradigm like E2E voting. These systems are radically different from existing systems that voters are accustomed to, the intuition behind certain procedural steps may not be immediately clear, and they have been known to confuse voters [516] [40] [402]. Furthermore, these systems task voters with the extra responsibility of vote verification, the necessity of which needs to be adequately explained and motivated. Dissemination of this instruction can take many forms, including programs to educate the general public about E2E voting systems via various media, supplying user-friendly instructions, public demonstrations of the technology and maintaining phone help-lines to walk voters through the voting and verification processes.

8.6 Conclusion

In this chapter we have undertaken a comprehensive introduction to the burgeoning field of end-to-end verifiable voting. We have traced the development of privacy and verifiability properties in the literature and described the workings of current state-of-the-art E2E voting systems. In our classification, we have attempted to track the intellectual evolution of these systems and highlight the key cryptographic and procedural techniques employed by their designers to harmonize security and usability concerns. We have also discussed current challenges to the deployment of these systems, consisting of outstanding technical, legal and usability issues.

Advances in E2E verifiable voting have the potential to restore trust in elections and improve democratic processes in society and, for that reason, we believe that the importance of these developments must not be underestimated. Our intention, in writing this chapter, has been to make the important innovations in this field accessible to a wider audience. We hope our work serves as a useful resource in this regard and assists in the future development of E2E voting.

Acknowledgments

The authors wish to thank Peter Hyun-Jeen Lee and Feng Hao for constructive discussions and Siamak F. Shahandashti, Jeremy Clark and Peter Ryan for helpful comments on the manuscript. This work is supported by ERC Starting Grant No. 306994.