Privacy and Verifiability in Voting Systems: Methods, Developments and Trends

Hugo Jonker[a,*], Sjouke Mauw[a,b], Jun Pang[a]

[a] *University of Luxembourg, Faculty of Sciences, Technology and Communication*
6, rue Coudenhove-Kalergi, L-1359 Luxembourg
[b] *University of Luxembourg, Interdisciplinary Centre for Security, Reliability and Trust*
4, rue Alphonse Weicker, L-2721 Luxembourg

Abstract

One of the most challenging aspects in computer-supported voting is to combine the apparently conflicting requirements of privacy and verifiability. On the one hand, privacy requires that a vote cannot be traced back from the result to a voter, while on the other hand, verifiability states that a voter can trace the effect of her vote on the result. This can be addressed using various privacy-enabling cryptographic primitives which also offer verifiability.

As more and more refined voting systems were proposed, understanding of first privacy and later verifiability in voting increased, and notions of privacy as well as notions of verifiability in voting became increasingly more refined. This has culminated in a variety of verifiable systems that use cryptographic primitives to ensure specific kinds of privacy. However, the corresponding privacy and verifiability claims are not often verified independently. When they are investigated, claims have been invalidated sufficiently often to warrant a cautious approach to them.

The multitude of notions, primitives and proposed solutions that claim to achieve both privacy and verifiability form an interesting but complex landscape. The purpose of this paper is to survey this landscape by providing an overview of the methods, developments and current trends regarding privacy and verifiability in voting systems.

Key words: Privacy, voting systems, receipt-freeness, coercion-resistance, end-to-end verifiability.

1. Evolution of privacy and verifiability requirements in voting systems

In a seminal paper Chaum proposed a technique to enable anonymous communication [Cha81]. In a single paragraph in this paper, he remarked on how such anonymous communications could be leveraged for private electronic voting. This remark triggered academic interest, which led to a plethora of voting system designs. These systems aimed

*Corresponding author. Tel: +352 466 644 5483, fax: +352 466 644 35483
Email addresses: hugo.jonker@uni.lu (Hugo Jonker), sjouke.mauw@uni.lu (Sjouke Mauw), jun.pang@uni.lu (Jun Pang)

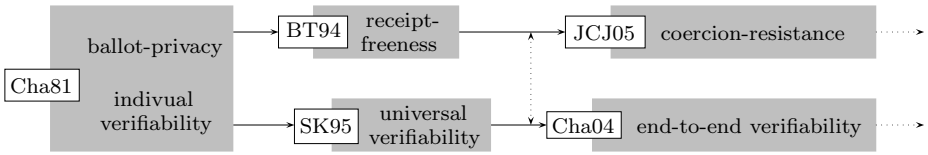


Figure 1: Evolution of privacy notions (top) and verifiability notions (bottom) in voting systems.

to achieve *ballot-privacy* – keeping the ballot's contents secret – while allowing voters to verify their vote counted towards the final result.

In order to ensure voter trust, most designs offered traceability of the voter's vote. Upon voting, the voter acquires a receipt, which she could use to trace her vote to the results. Benaloh and Tuinstra showed in [BT94] that such receipts can be used to nullify vote-privacy, when the voter shows her receipt to another party. They had the insight that vote-privacy is not optional, but must be enforced by the system. None of the classical systems that had been proposed up to then satisfied this requirement. BT94 led to a new direction in voting systems: *receipt-free* voting systems. These are systems that do not allow the voter to prove how she voted. Later, Juels, Catalano and Jakobsson [JCJ05] further refined the notion of privacy by identifying the need for *coercion-resistance*. This property extends the notion of receipt-freeness, by requiring protection against forced abstention, against forced randomised voting and against being forced to give up voter credentials. The evolution of these privacy notions is sketched in the top row in Figure 1.

Ensuring correctness of receipt-free and coercion-resistant systems is achieved by complex cryptographic operations. These operations also enable systems to satisfy *individual* and *universal verifiability*. Individual verifiability ensures that a voter can verify that her vote counts correctly, while universal verifiability (first recognised explicit by Sako and Kilian [SK95]) ensures that any observer (who may be a voter) can verify that the result is a correct reflection of the set of cast votes. However, the complexity of the cryptographic verification process hindered the adoption of theoretical voting systems in real-world elections. Practical and theoretical developments were not converging. Both Chaum [Cha04] and Neff recognised this as a problem and sought to address this, with the former's work setting a trend. Chaum's work [Cha04] ("Cha04" in Figure 1) took inspiration from both election practice and theoretical developments and proposed a voting system which offers a far easier approach (visual cryptography) to verify that the encryption of a vote does indeed contain that vote. Neff [Nef04] used a challenge/response protocol to achieve the same end. Their works led to a refinement of the notions of verifiability into *cast-as-intended* (as above), *recorded-as-cast* (the voter can verify that the encryption of the vote was properly stored), and *tallied-as-recorded* (anyone can verify that the announced result is correct w.r.t. the set of received votes). Chaum's and, to a lesser degree, Neff's works inspired a shift in focus, from systems focused on achieving strong privacy notions while being verifiable, to systems that focus on *practical* verifiability while satisfying strong privacy notions, such as receipt-freeness. Such systems are called *end-to-end verifiable* [Uni05] voting systems. The evolution of these verifiability notions is sketched in the bottom row in Figure 1.

All these developments led to a wide variety of proposed voting systems. However, a

current overview that describes these trends accurately, that is, that highlights firstly how strong notions of privacy were achieved, and later, how verifiability was made practical, is to the best of our knowledge missing. This survey investigates and clarifies the highlights of the above developments to provide such an overview. To that end, we distinguish two classes of systems along the lines of the developments sketched in Figure 1: systems focusing on privacy, and systems focusing on verifiability. We remark that privacy-oriented systems tend towards theoretical results (e.g., satisfying receipt-freeness), whereas verifiability-oriented systems additionally focus on practicality of implementation (e.g., satisfying receipt-freeness without requiring a voter to perform cryptographic computations). We consider a selection of voting systems, namely those proposed in recent years that notably offer an innovative approach to privacy and/or verifiability. We focus on the communication behaviour of each system, that is, which messages are exchanged by the involved parties. The systems are described in order of year of publication. The description of a system not only includes a sketch of how its salient features work and what claims it makes, but also refers to those studies (including non-security studies) we were aware of at the time of writing. Security breaks found by these studies are mentioned explicitly. Finally, we focus on democratic elections (e.g., voting systems to elect the government). Application areas such as boardroom-style voting are thus out of the scope of this survey.

Related surveys. In recent years, there have been few in-depth studies of developments in voting literature. Three notable exceptions are the studies by Sampigethaya et al. [SP05], by Smith [Smi05b], and by Cha et al. [CA12]. Smith's study focuses on the cryptographic blocks used to achieve privacy and verifiability. Cha et al. provide a brief survey of methods to determine the result of an election given the number of votes per candidate. Finally, the study by Sampigethaya et al. is closest of these three to our aim. They classify systems according to whether they hide voter, vote or both. Their study focuses on the cryptographic computations of the considered system.

In contrast, we set forth to sketch the recent developments and trends in voting literature, focusing on privacy and verifiability requirements. As such, we selected a group of papers that (in our view) have proven to be influential in the field, or added a novel approach towards privacy and/or verifiability. Moreover, in line with the intruder model due to Dolev and Yao [DY83], we focus on how communication flows and abstract away from the cryptographic details. The goal of our study is threefold:

(1) To help the reader understand various approaches to achieving privacy and verifiability in voting systems;

(2) To help the reader understand where these approaches originated, and how they developed further;

(3) To help the reader to make a high-level comparison of the discussed voting systems.

Structure. The survey is organised as follows: we begin with explaining how we view voting protocols in Section 2. In Sections 3 and 4, we review building blocks that are commonly used in voting systems to achieve privacy and verifiability, respectively. Next, voting systems are discussed. We distinguish between systems focused on achieving privacy goals from systems additionally striving for verifiability. In Section 5, we survey noteworthy voting systems focused on privacy, describing their operation and listing their

claims. This is followed by a survey of verifiability-oriented voting systems in Section 6. We conclude the paper in Section 7 with trends and developments observed in voting.

2. Terminology and notation

2.1. Terminology

As discussed above, this paper focuses on privacy and verifiability requirements. As remarked by Chevallier-Mames et al. [CFS+06], absolute privacy (no information leaked) cannot coexist with verifiability (sufficient information leaked to verify the result). The goal of voting system design is to find a way to enable as much privacy and verifiability simultaneously as possible. To this end, several terms for privacy and verifiability have arisen. In particular, we use the following terms:

Privacy:

- Ballot-privacy (BP): no outside observer can determine for whom a voter voted.
- Receipt-freeness (RF): a voter cannot prove *after the election* how she voted.
- Coercion-resistance (CR): a voter cannot interact with a coercer *during the election* to prove how she is voting.

Note that if a voter cannot prove how she voted, she must have had ballot-privacy (otherwise the empty proof suffices as such a proof). As such, we will view a claim of RF or CR as a claim of BP as well.

Verifiability:

- Individual verifiability (IV): a voter can verify that the ballot containing her vote is in the published set of "all" (as claimed by the system) votes.
- Universal verifiability (UV): anyone can verify that the result corresponds with the published set of "all" votes.
- End-to-end verifiability: a voter can verify that:
  - cast-as-intended (CAI): her choice was correctly denoted on the ballot by the system,
  - recorded-as-cast (RAC): her ballot was received the way she cast it,
  - tallied-as-recorded (TAR): her ballot counts as received.

Note that Kremer et al. [KRS10] identify another type of verifiability, *eligibility verifiability*: anyone can verify that each vote in the published set of "all" votes was cast by an eligible voter, and anyone can verify that each eligible voter cast at most one vote. Due to a lack of systems proposed in literature that claim to satisfy this requirement, we consider this type of verifiability not in scope for the current study.

Since the introduction of the notion of coercion-resistance, efforts have been made towards formal verification of privacy notions. This work has progressed along two lines. Firstly, there are efforts to model privacy properties in process algebras, such as in

applied π by Kremer and Ryan [KR05] and later Kremer, Ryan and Delaune [DKR06, DKR09], and in a specifically designed process algebra by Jonker, Mauw and Pang [JMP09b]. Verification using process algebras culminated in the successful automatic verification of protocols by Backes, Hriţcu and Maffei [BHM08]. Secondly, there are works formalising privacy in logics, including work by Jonker and De Vink [JdV06] and Bräunlich and Grimm [BG11] in first-order logic, and privacy in epistemic logic by Jonker and Pieters [JP06] and later Baskar, Ramanujam and Suresh [BRS07].

In addition to privacy and verifiability claims, there are a plethora of other security requirements for voting systems. For the sake of completeness, we list the claimed security requirements as well. Please be advised: we use the appellations the authors gave these, and, unlike for privacy and verifiability, make no attempt to harmonise terminology.

2.2. Notation

Encryption of a message m with key k is denoted as $enc_k(m)$. This ciphertext can be decrypted using the decryption key k^{-1}. In the case of symmetric cryptography, the encryption key k is equal to its inverse k^{-1}. For asymmetric cryptography, we denote the public key and private key of an agent A as $pk(A)$ and $sk(A)$, respectively.

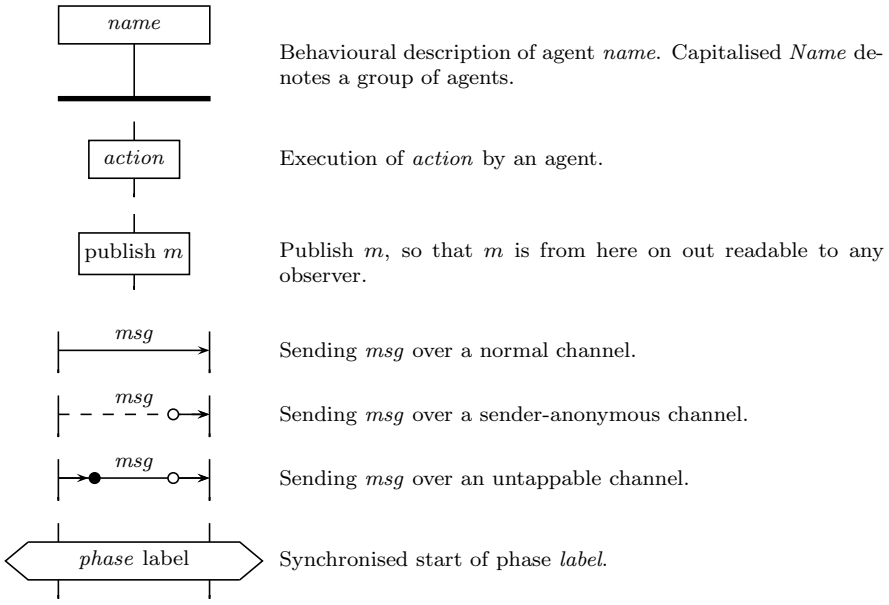


Figure 2: Legend for the interaction diagrams.

As an aid to comprehension, we provide interaction diagrams [MB01] for most of the privacy-oriented systems. The legend for these interaction diagrams is shown in Figure 2. From this legend, we emphasise that there exist a variety of actions, including the verification of signatures, validity checks, and publication of a message. Precise communication details concerning publication of a message are hardly ever given. To avoid suggesting or inadvertently introducting such details, we denote publication by one action.

Note that unless otherwise stated, all agents know their own private key and all other public keys (that is, key generation/distribution is not part of the diagrams).

3. Privacy building blocks

Several techniques have been introduced that enable electronic voting systems to provide privacy whilst preserving verifiability. In this and the next section we introduce some of the concepts that are prevalent in current voting systems. While we categorise these as privacy building blocks or verifiability building blocks, we stress that many of them specifically maintain privacy while allowing some form of verification, and can in fact be leveraged for both privacy and verifiability.

This section discusses building blocks for privacy. The building blocks here are homomorphic encryption, re-encryption, blind signatures, zero knowledge proofs, designated verifier proofs, secret sharing, mixnets, communication channels with privacy properties, and tamper-resistant hardware. Verifiability building blocks are discussed in the next session.

3.1. Homomorphic encryption

In a homomorphic cryptosystem, there is a homomorphism between an algebra based on the plaintext domain and an algebra based on the ciphertext domain. More precisely, using $\oplus$ to denote an appropriate operation on plain text messages and $\otimes$ to denote an appropriate operation on ciphertext messages, the following equality is the defining property of homomorphic encryption:

$$enc_k(m_1) \otimes enc_k(m_2) = enc_k(m_1 \oplus m_2).$$

This property allows two encrypted messages to be combined meaningfully (using $\otimes$) into one encrypted message without using a key. The actual operations represented by $\otimes$ and $\oplus$ depend on the used cryptographic system. Examples of homomorphic cryptographic systems include RSA [RSA78] (where both $\otimes$ and $\oplus$ are multiplication), ElGamal [ElG85] (where $\otimes$ is component-wise multiplication, and $\oplus$ is multiplication in the underlying group), and Paillier [Pai99] (where $\otimes$ is multiplication and $\oplus$ is addition). A detailed survey of homomorphic systems is given in [FG07].

Homomorphic encryption can be applied to tally votes using asymmetric cryptography. In such a setting, every voter v can encrypt her vote $vote_v$ with the election public key $pk(C)$, which yields $enc_{pk(C)}(vote_v)$. Using the homomorphic property of the encryption system, the authorities can add all votes together without opening the individual votes. This results in the encrypted tally:

$$enc_{pk(C)}(vote_1) \otimes \ldots \otimes enc_{pk(C)}(vote_n) = enc_{pk(C)}(vote_1 \oplus \ldots \oplus vote_n).$$

Anyone can verify the calculation of the tally and thus verify that her vote contributes to the result. However, no one except for the authorities that possess the decryption key can see for which candidate a particular voter voted. This preserves vote-privacy while offering verifiability.

There is one caveat with such processing of encrypted votes though: each encrypted vote must contain precisely one *valid* candidate. This is to avoid encryptions that contain

votes for two candidates, encryptions of a negative number (which translates as a "minus" vote for a candidate), and other such shenanigans. To ensure such fraudulent encryptions can be detected, each voter must also submit a *proof of validity* – a proof that her encrypted vote contains precisely one valid candidate (see e.g. [CGS97]).

3.2. Re-encryption

Re-encrypting a message means changing the ciphertext, without changing the corresponding plaintext. Knowledge of the decryption key is not needed for re-encryption, though the encryption key must be known. Recall that in symmetric cryptoschemes these keys are the same, so in any symmetric cryptosystem that allows re-encryption, the agent performing the re-encryption must possess the decryption key.

Re-encryption is related to probabilistic encryption, in which the ciphertext not only depends on the plaintext, but also on some random factor. More precisely, if a cryptosystem is not probabilistic, there is no random factor in the ciphertext that can be changed without changing the plaintext. We use $enc_{k,r}(m)$ to denote the probabilistic encryption of message m with key k and random factor r. A re-encryption of this message results in $enc_{k,r'}(m)$, where r' is the new random factor ($r' \neq r$). The re-encryption with random factor r of a given ciphertext c is denoted by $renc_{k,r}(c)$. A re-encrypted ciphertext is distinct from the original ciphertext. However, both decrypt to the same plaintext. The most relevant property of re-encryption is

$$renc_{k,r'}(enc_{k,r}(m)) = enc_{k,r''}(m),$$

where r'' denotes the random factor of the re-encrypted ciphertext, which depends only on r and r'. When the encryption key k is clear from the context, we will often write $renc_r(c)$ for $renc_{k,r}(c)$.

Examples of cryptographic systems allowing re-encryption include ElGamal [ElG85], Goldwasser-Micali [GM84], and Paillier [Pai99]. In fact, in any homomorphic cryptosystem which is probabilistic (that is, there are multiple encryptions c for every plaintext m), one can just homomorphically add encryptions of 0 to any ciphertext to obtain a different ciphertext without changing the plaintext.

In voting, re-encryption is used to break the link between a voter and her cast vote. By re-encrypting a cast vote, the vote remains unchanged, while the encrypted message can no longer be linked to the message sent by the voter while casting her vote.

3.3. Blind signatures

A party can authenticate a message by applying its digital signature to that message. Assuming an agent a's signing key $sk(a)$ and the corresponding public verification key $pk(a)$, we denote the signing of a message m by $sign_{sk(a)}(m)$. This signature can be verified with the operation $checksign_{pk(a)}$, satisfying the following equation:

$$checksign_{pk(a)}(sign_{sk(a)}(m)) = m.$$

If the keys and message are clear from the context, we will use *signature-ok?* to denote the verification of a signature by testing the above equality.

A *blind* signature [Cha82] is a way for a party to sign a message without knowing the message. Compare this to a sealed envelope, containing a piece of carbon paper and

a message to be signed. The carbon paper will copy anything written on the envelope to the message, including a signature. Thus, a notary can sign the envelope, while the secret message is not revealed to him. The envelope (the blind) can later be removed, revealing a signed message.

We denote a message m blinded with blinding key b by $blind_b(m)$. If a blinded message is signed, the message can be unblinded without removing the signature. For a message m, signed with agent a's key $sk(a)$, we have the following:

$$deblind_b(sign_{sk(a)}(blind_b(m))) = sign_{sk(a)}(m).$$

Blind signatures can be used to ensure that only eligible voters can vote (see Figure 3; the used notation is explained in Figure 2). A voter v blinds her vote $vote_v$ using blinding key b. She signs that and sends her signed, blinded vote to the registrar. The registrar separates the blinded message and the signature, and verifies that the voter is eligible and has not yet voted. If so, the registrar signs the blinded vote and sends it back to the voter. The voter unblinds the received message using b and sends the signed vote to a counter.

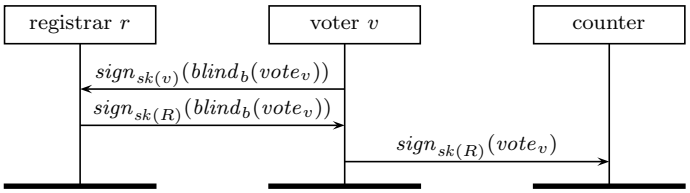


Figure 3: An example of the use of blind signatures.

3.4. Zero knowledge proofs

A zero knowledge proof (ZKP) is a proof of a statement s which reveals nothing about statement s, except that it is true. An informal example is the following: the prover sets out to prove to a color-blind friend that two billiard balls have a different color. The friend takes one ball in each hand, showing the prover which ball is in which hand. Then, the friend puts the balls behind his back and may or may not switch them. Finally, the friend shows the balls again, and the prover has to tell whether the balls were switched.

In the example, if the balls are truly indistinguishable, the prover only has a 50% chance of guessing correctly. Therefore, by repeating this process a few times, the friend can reduce the chance of the prover guessing correctly as much as he wishes.

Remark that in the example, a zero knowledge proof deals with some public knowledge and some private knowledge. In the example, the order of the balls in the beginning is public knowledge, while the knowledge of whether they switched hand is private. In general, a zero knowledge proof proves a boolean statement over publicly and privately known arguments, without revealing the private knowledge.

A common type of zero knowledge proofs is based on a commit-challenge-response cycle: the prover commits to a certain solution based on his secret knowledge, the verifier challenges the prover, and the prover reveals a part of the commit, showing it is consistent with the prover's claim. With repeated execution the verifier becomes more and more assured that the claim is true. Zero knowledge proofs based on such interactions are called

interactive. Fiat and Shamir [FS86] proposed a method to transform such interactive proofs into *non-interactive* zero knowledge proofs. This relies on the prover generating a (to him) unpredictable challenge based on his commitment and public knowledge, such as the hash of the commitment. Since this is unpredictable, the prover cannot tweak the commitment to give him an advantage.

Zero knowledge proofs are often used in voting to prove to a voter how her vote is being processed by the system. This enables a voter to verify that her vote is being handled correctly, without her being able to prove to an outsider how she voted. An example of this is proving a certain ciphertext is indeed a correct (re-)encryption of a message, without revealing the encryption key. This is done in mixnets (see Section 3.7).

3.5. Designated verifier proofs

Regular non-interactive proofs are transferable, i.e., a voter who possesses a proof of a boolean statement can transfer that proof to anyone. This is not always desirable, for example a voter selling a proof of how she voted. Suppose prover P wants to prove the truth of statement s to verifier V, but doesn't want V to be able to prove s to anyone else. What P can do, is prove the following to V:

"I am V OR s."

As the verifier knows that the prover is not V, the statement s must be true. As "I am V" holds true for the verifier, the verifier can prove "I am V OR x." for any x (by proving that the verifier is V). Hence, V cannot use the above proof to convince anyone else of statement s. Such a non-transferable proof is called a designated verifier proof [JSI96].

3.6. Secret sharing

Secure operation of a system can often only be guaranteed under the assumption that one of the agents is trusted. An authority in a voting system is an example of such a trusted agent. However, access to information is sometimes deemed too sensitive to be left to one agent. Compromising a single agent then would suffice for an attacker to break the system's security. To avoid this, cryptographic mechanisms have been developed that allow the sharing of a secret by a number of authorities. This is done in such a way that only by putting their shares together, the authorities can recover the secret [Sha79, Bla79].

A simple way to share a secret is to encode the secret into a bitstring, and choose the shares such that the *exclusive or* of all shares produces the secret. While this works, this requires each agent holding a share to be honest about their share. Failing one share, the secret becomes unrecoverable.

To counter this, more complex secret sharing schemes have been proposed. One method works by interpolation of polynomials (see Figure 4). To share a secret over n parties, the secret is encoded as a number m. Then, a random polynomial of degree t is chosen that covers the point $(0, m)$ (in Figure 4: $-0.2x^3 + 0.5x^2 + 0.7x + 1$). Each share is then given by a random point on the polynomial. The example shows a polynomial of degree three. There are four shares, since a polynomial of degree three is fully determined by four points.

To avoid a block by any one share holder, a random polynomial of lower degree can be used (e.g. of degree 2) to share the secret. In this case, only three of the four shares

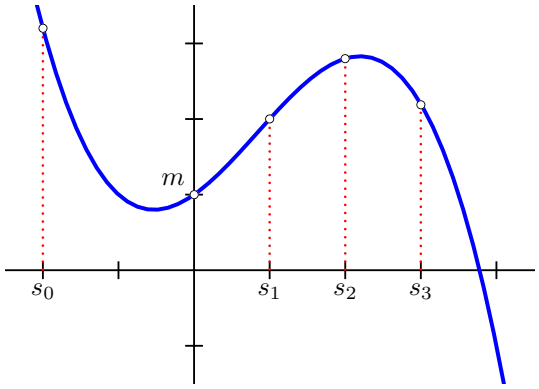


Figure 4: Secret m shared over s_0, s_1, s_2 and s_3 using polynomial $-0.2x^3 + 0.5x^2 + 0.7x + 1$.

are needed to recover the secret. This is called *threshold cryptography*. Often, a secret sharing scheme is referred to as (t, n) meaning that t (for threshold) shares are needed, and that there are n shares in total.

Secret sharing can be used in voting to distribute a decryption key. Suppose all voters encrypt their votes with the same public key. The corresponding private key is then split over all authorities, to ensure that only the authorities acting in concert can decrypt the votes. In some cryptosystems such as ElGamal, it is even possible to decrypt a ciphertext without reconstructing the key – see e.g. [CGS97].

Example of decryption without reconstructing the key. Encryption in ElGamal works as follows. The public key is given by raising the generator g of a given group to the private key k: g^k. Encrypting message m results in $(g^r, (g^k)^r m)$, where r is chosen by the encrypting agent. To decrypt, one raises the first element g^r to the power k, which results in the term g^{rk}. Then, the second element $(g^k)^r m$ is divided by g^{rk}, resulting in m. Note that the private key k is not needed for decryption. Decryption only requires g^{rk}, which is derived from k.

A joint decryption of $(g^r, (g^k)^r m)$ without reconstructing the key transpires as follows. Assume the private key k is split into n shares $s1, \ldots, sn$, such that $g^k = g^{s1+s2+\cdots+sn}$. Each shareholder i raises g^r to his share and sends $(g^r)^{si}$ to the others. Now, the shareholders can compute $\Pi_i (g^r)^{si} = (g^r)^k$, which is the term needed to decrypt $(g^r, (g^k)^r m)$. In this fashion, the message can be recovered without constructing k.

3.7. Mixnets

A simple way to achieve privacy of one's vote is to shuffle it with a number of votes from other voters. This can be achieved by the use of *mixes* [Cha81]. A mix shuffles its input in such a way that no element of the output can be linked back to an element in the input. To ensure that the link between the input list and the output list is not trivially retrieved, the mix needs to change the appearance of every message *without* changing the meaning of any message.

There are basically two ways in which the shuffling can be implemented, viz. using *re-encryption mixes* and using *decryption mixes*. A re-encryption mix works as follows. To shuffle a list, a mix receives encrypted elements, re-encrypts them, and randomises

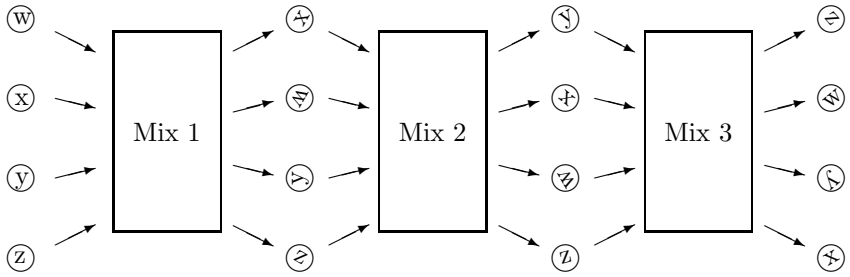


Figure 5: An example of a mixnet in operation (adapted from [HS00]).

the order in which the elements are output. In order to ensure that no single entity can recover the ordering, multiple mixes can be chained to form a *mixnet*. The final output elements then can only be linked to their corresponding input elements if all mixes collude.

When using decryption mixes, the initial input consists of several layers of encryption, each layer specifically targeted at one mix. On its way through the mixnet, the original message changes its shape because each mix peels off one layer of encryption. This approach is used in onion routing [SGR97], which provides anonymous routing.

Figure 5 sketches the working of a mixnet. The first mix takes an input list of messages (here, w, x, y, z) and outputs a shuffled list (x, w, y, z) in such a way that the individual messages cannot be linked to its input (denoted in Figure 5 by rotation of the messages). The output of the first mix becomes the input of the second mix, which in turn provides the input to the last mix. The output of the last mix is not linkable to the input of the first mix, nor to any of the intermediate stages.

To prove that the output is indeed a permutation of the input, the mix can use zero-knowledge proofs or designated verifier proofs. A pair of mixes has another option available, called randomised partial checking [JJR02]: they shuffle and publish their results. An independent party then selects for each message in the intermediate set (i.e., the output of the first mix, which is the input of the second mix) whether the link to the input or the link to the final output is revealed. A variant of this is shown in Figure 25. As the mixes do not know in advance which link they must reveal, the likelihood of being caught should they mix fraudulently is substantial. At the same time, input messages cannot be linked to the output messages as only one of the two links necessary for linking is revealed for any message.

Mixnets are used to provide anonymity in various ways in voting systems. One approach is to use mixnets to construct an anonymous channel (a communication channel that hides who put a message on the channel). Another use is to shuffle the list of candidates on a specific ballot. The order of the candidate is then revealed to the voter (using a designated verifier proof) and correctness of the shuffle is proven publicly (using zero knowledge proofs). A third option, proposed in [Nef01], is to mix voter credentials together, such that the voters are no longer individually identifiable, but only as a group.

3.8. Anonymous and untappable communication channels

To ensure privacy, many voting systems rely on communication channels with special properties: *anonymous* channels and *untappable* channels. Sender-anonymous channels hide the sender of a message, and can be implemented using a mixnet as first suggested by Chaum [Cha81]. An untappable channel is a channel where no party except sender and receiver can learn anything about the communication, including whether communication occurred or not. One approach to achieve this, is to offer multiple channels, where the sender picks one at random. Under the assumption that an adversary cannot tap all channels simultaneously, the adversary cannot be certain that none of the channels was used. Nevertheless, an untappable channel remains a very strong physical assumption. Replacing this with less strict assumptions (such as by a threshold homomorphic deniable encryption scheme as suggested by [CGS97]) remains currently an open problem.

3.9. Tamper-resistant hardware

Another approach to achieving privacy is to rely on special hardware, designed so that it is very difficult to modify or inspect the internal functioning of the hardware (tamper-resistant). An everyday example of this is a smart card. Given the assumption of tamper-resistance, the hardware can be viewed as a trusted computing base whose computations can neither be inspected nor disturbed. Voting systems can leverage this to re-encrypt a vote, ensuring that the voter cannot prove how she voted by decrypting her vote.

3.10. Security implications of cryptographic primitives

The cryptographic primitives discussed above depend on specific assumptions and may introduce their own limitations. For example, homomorphic encryption, re-encryption (and thus re-encryption mixnets) and blind signatures each allow a ciphertext to be altered in a meaningful way without access to the decryption key. Cryptographic systems with this property are called *malleable* [DDN00]. Malleability of the cryptographic system can be leveraged for privacy, because ciphertext(s) can be meaningfully changed (e.g., adding two ciphertexts together) by someone who does not know the plaintext(s). However, a malleable ciphertext can be changed in a meaningful way by an attacker as well. Therefore, such cryptosystems must be used with care and awareness of their limitations and vulnerabilities.

Moreover, most of the cryptographic primitives are *computationally secure*. This means that security relies on a mathematical problem (such as factorisation, or discrete log) believed to be intractable to solve. In contrast, a primitive that cannot be broken even when the adversary has unlimited computing power is called *information-theoretically secure*.

Finally, remark that a *deterministic* cryptosystem – that is, a cryptosystem such that for every plaintext p there is precisely one ciphertext c – will trivially reveal who voted for whom. The attacker simply encrypts each possible vote, and so acquires a mapping from each candidate to his ciphertext. The attacker can then use this mapping to decrypt any ciphertext. Therefore, any cryptosystem used in voting must be non-deterministic.

4. Building blocks for verifiability

The building blocks for verifiability discussed below are secure bulletin boards, ballot auditing (cast-or-audit, continuous auditing, Markpledge-style auditing), plaintext equivalence tests, and commitment schemes.

4.1. Bulletin boards

Many voting systems from literature assume the use of secure bulletin boards. Typically, this is used to provide a verifiable log of communications. To this end, a secure bulletin board provides an append-only, publicly available communication medium with memory. The basic properties (see e.g. [JP11]) are:

- information cannot be removed or modified, only added to the board;
- anyone may read the board,
- the board provides a consistent view to everyone.

Most systems discussed in this survey state that there is a bulletin board, but not how to communicate with it. To avoid suggesting any specific type of interaction, in the interaction diagrams the act of publishing is denoted by an action "publish " (cf. Figure 2).

4.2. Ballot auditing and ballot verification

To preserve privacy, many systems opt to record a voter's choice in an encrypted way. As Chaum [Cha04] pointed out, there should be some mechanism to ensure the voter that the encryption indeed contains the voter's choice. This can be achieved with cryptographic proofs (e.g., visual cryptography as Chaum proposed, zero-knowledge proofs or designated-verifier proofs), but such proofs cannot convince voters who are not cryptographic experts. In literature, three alternative approaches have been proposed: verification by means of short strings (Markpledge), continuous auditing (Prêt à Voter), and cast-or-audit (Benaloh challenges).

Verification using short strings. Neff [Nef01] proposed the Markpledge system (Section 6.5), that uses short strings to enable voters to verify that the ballot posted publicly was the one they saw in the voting booth. While full verification relies on helper organisations, this approach does put some power in the hands of the voter as follows: the voter makes her choice for candidate A. Next, the voting system prints a ballot that contains encryptions for all candidates. The machine prints a commitment to the encryption of the chosen candidate. The voter inserts her challenge (a short string), and the machine prints the challenge and the answer for the chosen candidate. Finally, the machine prints a cryptographic proof that the printed receipt contains precisely one "yes" vote. (To preserve privacy, the machine also computes fake commitments matching the challenge and prints fake answers for the other candidates.)

As the challenge string is short, the voter can easily remember it. This lets the voter easily verify that the challenge printed is indeed the one she put in. In addition, she can take the printed receipt to a helper organisation, who verifies the cryptographic details (there is only one "yes" vote, and all the answers match the commitments and the challenge).

Continuous auditing (Prêt à Voter). Ryan [Rya05] suggested to have continuous audits of the ballot encryptions in the Prêt à Voter (Section 6.6) system. In Prêt à Voter , the authorities prepare ballots in which the candidate order is randomised, and encrypted. To be sure that the order that is claimed is indeed the order that is encrypted, voters can submit a blank ballot and have that ballot's order decrypted, spoiling the ballot. After repeated testing, the voter is convinced that the candidate order shown on the form is the one encrypted. At this point, the voter selects a random blank ballot, marks her choice and casts her vote.

Cast-and-audit (Benaloh challenges). Benolah [Ben06, Ben07] elaborated this auditing approach to a system where the voter marks her choice, the system prepares the ballot, and then the voter is presented with the option to either decrypt the prepared ballot, or to cast the prepared ballot.

The paper coming out of the voting device can now act either as a ballot or as an auditing tool. Since the voting device is irrevocably committed to a particular encryption, and as the device cannot predict whether the voter will choose to cast or choose to audit, any cheating by the voting device has a 50% chance of being audited (and, thus, detected). By repeating the audit as often as desired, the voter can test the machine as often as desired and thus increase her confidence in the correctness of the machine's operation.

4.3. Plaintext equivalence tests

A plaintext equivalence test (PET) is a test of two ciphertexts, that results in true if the two respective plaintexts are equal, without revealing anything else. In other words,

$$PET(\{ma\}_k, \{mb\}_k) \quad \equiv \quad ma = mb.$$

Such tests are used in various systems. For example, the JCJ05 system uses such tests to identify and remove duplicate votes from the bulletin board without revealing anything about the duplicated votes.

4.4. Encoding candidates

To ease the tallying process, candidates can be encoded as numbers, while the operator $\oplus$ represents addition. If we assume, for instance, 3 candidates and 9 voters, then a vote for the first candidate is represented by 1, a vote for the second candidate by 10, and a vote for the third candidate by 100. The addition of nine such votes then results in a number with at most 3 digits that represents the outcome of the voting process. For instance, if the sum of the votes equals 315, then there were 5 votes for the first candidate, 1 for the second and 3 for the third. The generalisation of this encoding to more voters requires the numbers to be represented in a different base. Such an encoding is due to Baudron et al. [BFP$^+$01], and can be combined with homomorphic tallying to enable anyone to verify correctness of the result.

Note that if the encoded votes are encrypted, care must be taken to ensure only correctly encoded votes are accepted (see the remark on processing encrypted votes in Section 3.1).

4.5. Commitment schemes

In proofs, it is often necessary that the prover first commits to a certain statement m (e.g., the candidate order) and later reveals that commitment. Given a commitment key c, which is normally a random string chosen by the prover, the main property of a commitment scheme is:

$$\text{decommit}_c(\mathit{commit}_c(m)) = m.$$

Whenever the commitment key c is not relevant for understanding the protocols, we will leave it out. Another property of commitment schemes is that the prover cannot deny this commitment later, that is, the prover cannot find m', c' such that

$$\text{decommit}_{c'}(\mathit{commit}_c(m)) = m'.$$

This property can be achieved using regular encryption to commit, and afterwards revealing the decryption key. However, encryption schemes are by nature only computationally hiding: given enough computational resources, the decryption key will be found. Commitment schemes (e.g., [Ped91]) can be "perfectly" hiding: *no* amount of computation can decide which value was committed to. This can be achieved, for example, because the commitment can be opened in any way:

$$\forall m' \; \exists c' \;\; \text{decommit}_{c'}(\mathit{commit}_c(m)) = m',$$

but to find such a m', c' pair is computationally hard. Thus, while the prover cannot find such an alternative pair in time to lie about his commitment, anyone trying to break the commitment later will find that the commitment could have been to any message. Several voting systems (e.g., [CFSY96, MN06, MN07]) let the voter cast commitments instead of encryptions. For these systems, even if the commitments are made public and linked to the casting voter, no one can ever find out how a voter voted.

Trapdoor commitments. An interesting variety of commitment schemes are trapdoor commitment schemes. In these schemes, knowledge of a trapdoor does allow the committer to open the commit in any way. To assign her vote to a candidate, the voter sends the decommit value c privately to the election authorities. Using this decommit value, the commit will open to the chosen candidate. But for every other candidate m', the voter can compute an appropriate c' such that the commit can be opened as if it is a vote for m'. In this way, the commits to candidates can be made public, without the voter being able to prove which candidate she committed to. This approach was used in a.o. [Oka96, Oka97].

5. Survey of voting systems focusing on privacy

This section surveys the class of voting systems focusing on privacy, with a particular focus on receipt-free and coercion-resistant systems. In line with the goal of this paper, this section focuses on the properties of and relations between the most noteworthy, interesting, and innovative systems. As is common, systems built upon the ideas and concepts of their predecessors. In Figure 6, we sketch the stronger of such influences – tracing those ancestor systems whose ideas contributed significantly to a particular system. There is one exception: to maintain legibility, we did not trace the influences of Chaum's initial paper ("Cha81" in the figure).

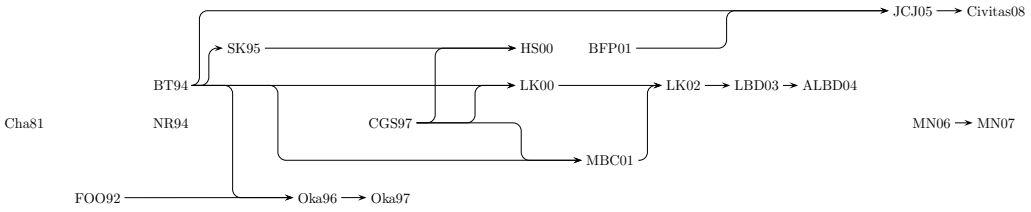


Figure 6: Influences between notable voting systems focussed on privacy.

5.1. Overview

There is a vast body of voting literature concerned with ballot-privacy. Most works published after the seminal work of Benaloh and Tuinstra (BT94) address their stronger privacy requirements. The systems are discussed in order of publication year.

Naturally, design of receipt-free systems is influenced by their (non-receipt-free) predecessors. Below we discuss two influential predecessor systems: the FOO92 system [FOO92] and the CGS97 system [CGS97]. The FOO92 system was designed with the intent of actual use. It has been used as the basis for implementation of two voting systems (Sensus [CC97] and Evox [Her97]). CGS97 was designed with the intent of being as efficient as possible. As such, CGS97 has been used as a basis by various theoretical systems (including HS00, LK00, MBC01) that aim to incorporate receipt-freeness.

But the main focus is on receipt-free systems. We discuss the seminal work by Benaloh and Tuinstra (BT94) [BT94]. This is followed by an alternative proposal by Niemi and Renvall (NR94) [NR94], which appeared around the same time and addressed the same concerns. From then on, research into voting became quite prolific, and we highlight the influential receipt-free systems that followed their work: Oka96/Oka97, HS00, MBC01, BFP01. In addition, to illustrate a typical development cycle, we highlight the sequence of systems LK00, LK02, LBD03, ALBD04.

After discussing receipt-freeness, we outline JCJ05, the work of Juels, Catalano and Jakobsson [JCJ05]. Their introduction of the notion of coercion-resistance has led to a new direction in voting. Then we examine the MN06 and MN07 systems by Moran and Naor [MN06, MN07] on achieving unconditional privacy – at the cost of having computational (instead of unconditional) verifiability. Finally, we discuss Civitas, an implementation of JCJ05 with some modifications. We end our survey of privacy-focused systems here, as we find that the focus of the community began shifting. The community is steadily moving towards combining existing understanding of privacy with *practical* verifiability, and away from further refining privacy.

5.2. FOO92: anonymity using blind signatures

The FOO92 system [FOO92] by Fujioka, Okamoto and Ohta aims to combine privacy with verifiability. To achieve this, it uses blind signatures as follows (see Figure 7). The voter encrypts her vote (using a commitment scheme), blinds that encryption and sends the blinded, encrypted vote to the registrar (the first message of Figure 7). The registrar verifies that the voter is an eligible voter and that she hasn't voted yet. If so, the registrar signs the blinded, encrypted vote and sends the signed version to the voter (2^{nd} message). The voter then unblinds this message and obtains an encrypted vote, signed by the registrar. She sends this via an anonymous channel to the counter (3^{rd} message).

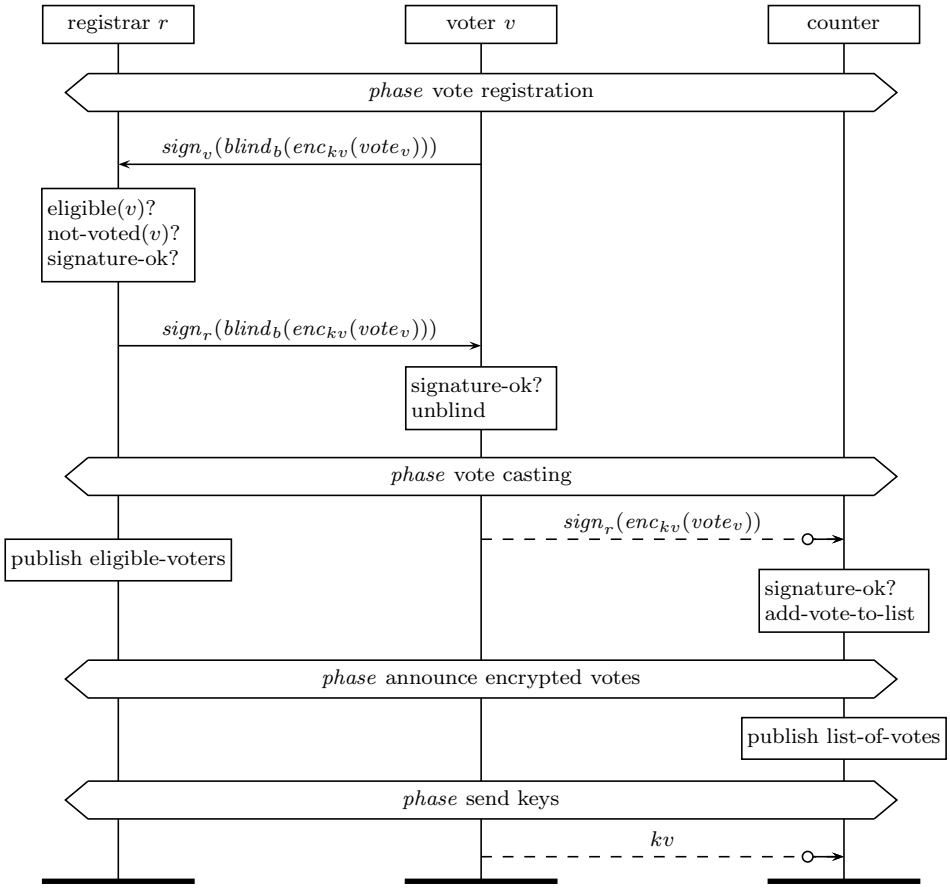


Figure 7: The FOO92 system.

Then, after the voting period has ended, the counter publishes the list of all signed, encrypted votes that he received. After publication, the voter sends the decryption key for her vote to the counter (the last message in Figure 7). This allows the counter to open the vote and count the votes.

FOO92 serves as the basis for two implemented voting systems: *Sensus* [CC97] and *Evox* [Her97]. Evox implements the anonymous channel as a trusted third party; Sensus omits the anonymous channel completely. As such, both implementations deviate from the FOO92 specification in a manner that negatively impacts voter-privacy.

The FOO92 system has some peculiarities avoided in later voting systems. For example, all voters must take action twice, once to cast their votes and once to send the used encryption key. Furthermore, the system predates the introduction of stronger privacy requirements. Consequently, it is not receipt-free: the decryption key of a particular vote constitutes sufficient receipt [JdV06]. As such, the FOO92 system is not considered secure, even though it may satisfy all its original security goals.

Claims. FOO92 claims ballot-privacy (BP) and an early form of universal verifiability (UV). Moreover, it makes the following additional security claims: completeness (all votes counted correctly), soundness (malformed votes cannot change the result), no double voting, eligibility, and fairness.

Studies. Privacy of the FOO92 system is studied in a.o. [KR05, MVdV07, JdV06, BRS07], fairness in a.o. [KR05, BRS07, TMT+08], eligibility in a.o. [KR05, TMT+08], and verifiability of FOO92 is studied in a.o. [BRS07, TMT+08, KRS10].

5.3. BT94: introducing receipt-free voting

In their landmark paper, Benaloh and Tuinstra [BT94] described an attack on vote-privacy, apparently used in Italy. In certain election systems used for multiple parallel elections in Italy, voters could list their votes in any order. Knowing this, and given that the number of possible permutations is far greater than the number of voters, an attacker can assign specific permutations of votes a priori. This permutation then acts as a voter's "signature". If a voter's signature is not in the set of cast votes, the voter deviated from her assigned voting strategy. If the signature is present, then the voter complied with her assigned voting strategy.

The above example illustrates two important things. First, it exemplifies that if the voter can somehow undo the privacy of her vote, she is at risk of being coerced. Secondly, it also shows that *any* input of the voter may allow her to lift her privacy.

Benaloh and Tuinstra devised a way around this problem. Their solution is constrained to voting for either 0 or 1. The solution relies on a "doubly" homomorphic randomised cryptosystem, that is, an encryption system that supports two homomorphic operations: ciphertext operations $\otimes$ and $\oslash$ with plaintext equivalent operations $+$ and $-$, respectively. Moreover, this cryptosystem must be able to generate proofs of decryption.

The central idea is to use an interactive challenge-response protocol, where only the voter knows the commitment (in Figure 8, the claimed values of the decryptions of a_i). As such, only the voter is convinced. More specifically, the voting authority publishes a series of pairs $(a_i, b_i), 0 \leq i \leq N$, containing an encrypted 0 and an encrypted 1 in random order. The authority then privately commits to the decryption of the published pairs. Next, an independent and trustworthy random source ("beacon" in the figure) generates the challenge c. The authority responds to the challenge (see below), after which the voter knows which encryption contains her preference. She publicly announces which encryption (either a_0 or b_0) she prefers. Finally, to determine the result, the authority sums up all encrypted votes using the $\oplus$ homomorphic encryption, and decrypts the result.

The authority responds as follows: If the challenge bit $c_i = 0$, the authority shows that (a_i, b_i) is decrypted as either $(0, 1)$ or as $(1, 0)$. Conversely, if the challenge bit $c_i = 1$, the authority proves it knows which value is which in (a_0, b_0), by homomorphically combining the components with those of (a_i, b_i) into one pair such that one pair is $(enc(0), enc(0))$. Thus, the authority proves both that (in case $c_i = 1$) the pair (a_0, b_0) contains the same plaintext (though perhaps differently ordered) as the other pairs, and that (in case $c_i = 0$) the other pairs consist of an encrypted 0 and an encrypted 1. Therefore, (a_0, b_0) must also consist of an encrypted 0 and an encrypted 1. Moreover, the voter knows which is which and can thus select her preference and publicly announce it.

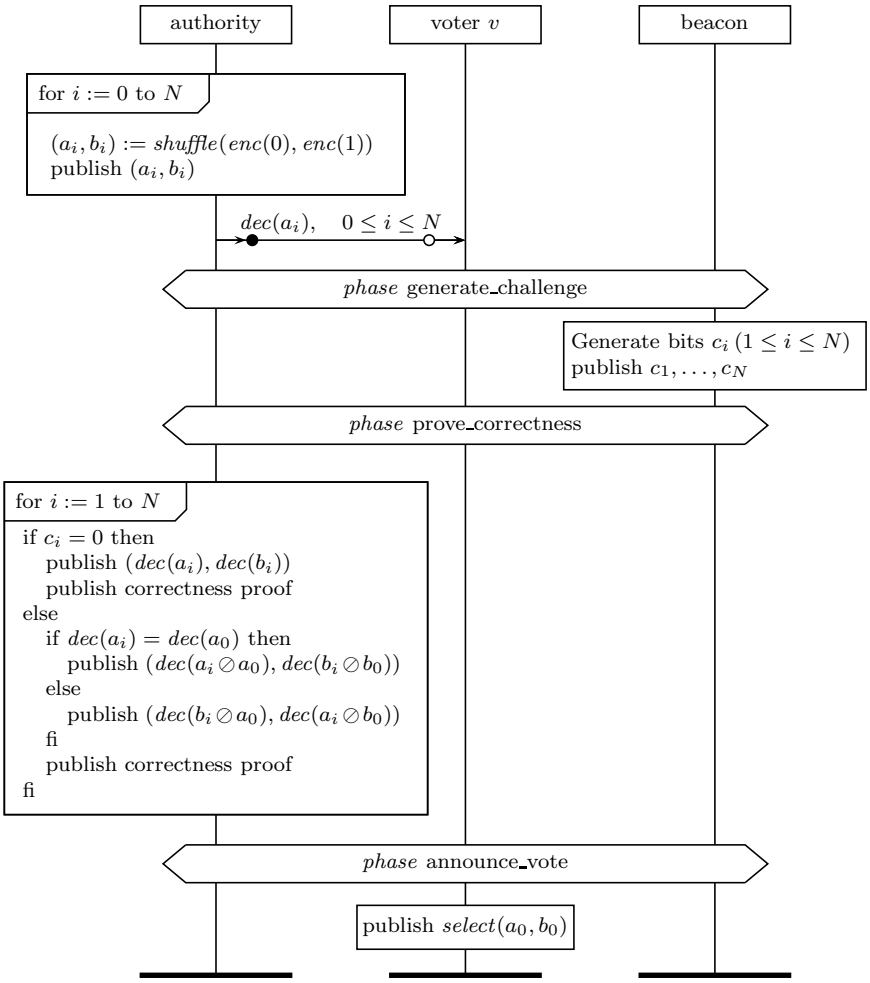


Figure 8: Vote casting in BT94

In addition to requiring a cryptographic system with very specific properties, the BT94 system relies on the existence of a voting booth, and a private channel. While cryptographic systems satisfying the particular requirements exist (e.g. [Ben94]), Benaloh and Tuinstra left it unclear how to satisfy these latter two assumptions. In addition, their system can only handle a maximum of two candidates. Schemes following BT94 sought to address these shortcomings, as described below.

Claims. BT94 claims receipt-freeness (RF), and does not explicitly claim a verifiability property. In addition, it claims "correctness of result".

Studies. Hirt and Sako [HS00] identified how to generate a receipt in the multi-candidate variant of BT94. Jonker et al. [JdV06] analysed privacy of the two-candidate variant. Fousse et al. [FLA11] discovered that in Benaloh's homomorphic encryption scheme [Ben94]

two different plaintexts are sometimes mapped to the same ciphertext. They proposed a fix which extends to BT94.

5.4. NR94: unlinkable receipts

Around the same time, Niemi and Renvall [NR94] independently identified the threat of vote buying as well. They also noted that all existing systems gave the voter a token, which is made public upon counting. Their solution was to adapt this token, so that the voter cannot prove that it is hers. The (multi-party) computations done for this occur inside a voting booth, which keeps the voter completely private. The token allows a voter to check if her vote is part of the public set of received votes (ensuring individual verifiability). Furthermore, the correctness of the count can be publicly verified (ensuring universal verifiability).

As in the receipt mentioned for FOO92 [JdV06], a voter can predict a random value that will appear on the bulletin board. However, this is not a receipt as the voter may generate fake tokens and thus cast fake votes, which are indistinguishable from real votes. (Note the similarity between this approach to receipt-freeness and the ideas to ensure coercion-resistance in JCJ05.)

The computations needed do not scale well with respect to false votes, something which Niemi and Renvall already noted. Furthermore, the system requires a strict assumption (a voting booth). Although their proposal was influential for its ideas (like [BT94]), the mechanics they proposed have been relegated to the fringes of voting research.

Claims. NR94 claims receipt-freeness (RF), and does not explicitly claim a verifiability property.

5.5. Oka96, Oka97: receipt-free blind signatures

Okamoto proposed an adaption of FOO92 to achieve receipt-freeness [Oka96] (see Figure 9). The main differences with FOO92 (as depicted in Figure 7) are: the direct and anonymous publishing (by voters) on the bulletin board ("anon-publish" in Figure 9), the use of anonymous untappable channels to the authorities, and the use of a trapdoor commitment scheme to encrypt the votes (denoted as $commit_{kv,n}$ in Figure 9). Knowledge of the trapdoor (r in Figure 9) allows an encryption to be opened in any way (i.e. for any candidate). Due to this, the encrypted vote does not constitute a receipt.

However, Okamoto noted a flaw in the system: the trapdoor is voter-generated (see also [MBC01]). An intruder can force a voter to use an intruder-generated trapdoor. Then, the voter can still encrypt her vote, but she can only open this encryption in one way. In [Oka97], Okamoto proposed three variations to address this flaw. The first uses secret sharing to break up trapdoor r into shares for every counter. The second system generalises the secret sharing to a threshold secret sharing scheme. The third uses a "voting booth" (here, a two-way untappable channel), which the voter can use anonymously. The voter then proves via the voting booth she knows r, thus preventing the attack on Oka96. We refer to the three systems collectively as Oka97.

Claims. Both of the Oka96 and Oka97 systems claim receipt-freeness. Oka96 does not explicitly claim verifiability properties, but does additionally claim fairness and privacy. Oka97 claims that blind signature schemes with a public bulletin board inherently satisfy both individual and universal verifiability.

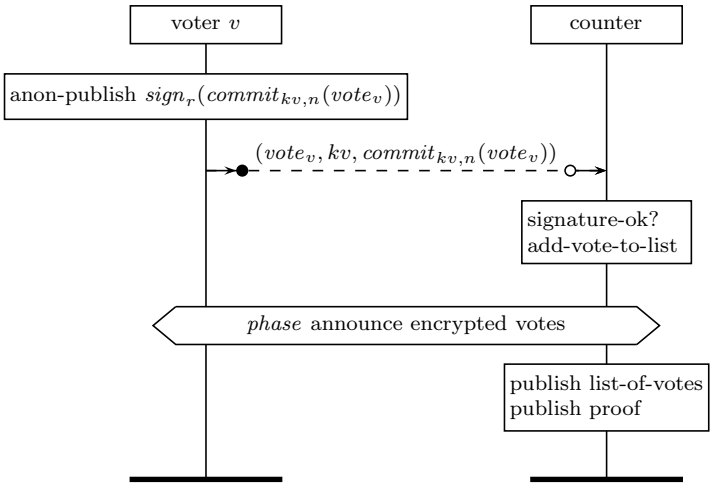


Figure 9: The Oka96 vote casting phase (improving FOO92).

Studies. Privacy of Oka97 was studied in a.o. [KT09]. Delaune et al. [DKR09] gave a formal analysis of Oka96, finding receipt-freeness satisfied and coercion-resistance violated, confirming Okamoto's own findings [Oka97]. Dreier et al. have similar results [DLL11, DLL12]. Juels et al. [JCJ05] remarked that Oka97 achieves resistance to forced abstention, by verifying eligibility only in private – which carries its own problems, according to Juels et al.

5.6. CGS97: optimally efficient voting

The system proposed by Cramer, Gennaro and Schoenmakers [CGS97] aims to be optimally efficient for a multi-authority setting. This work builds on efforts by Cramer and others [CFSY96] on improving efficiency in a multi-authority setting. A multi-authority setting is desirable, as it reduces the trust assumption required for each individual authority. In their system, a voter publishes a single encrypted vote plus a proof that the published message contains a valid vote. The system is a homomorphic (ElGamal) voting system, it sums all encrypted votes and then uses joint decryption to decrypt the sum. Cramer et al. noted that proofs of validity of ciphertexts (as discussed in Section 3.1) can be made non-interactive using the Fiat-Shamir technique [FS86]. They remark that in this case, care should then be taken to ensure that the constructed challenges are voter-specific (to avoid duplications of challenges). The system is sketched in Figure 10, where $pk(C)$ denotes the public key shared by the counters. Remark that all communication happens via the bulletin board. In particular, the jointly-decrypt-sums action is run via the bulletin board, and leaves any observer able to verify correctness of the announced tally.

Cramer et al. are aware that their system is not receipt-free. They stated that this can be achieved using special channels, which in their view could be replaced with tamper-resistant hardware. They raised the idea of receipt-freeness without untappable channels, similar to incoercible multi-party computation without untappable channels. However,

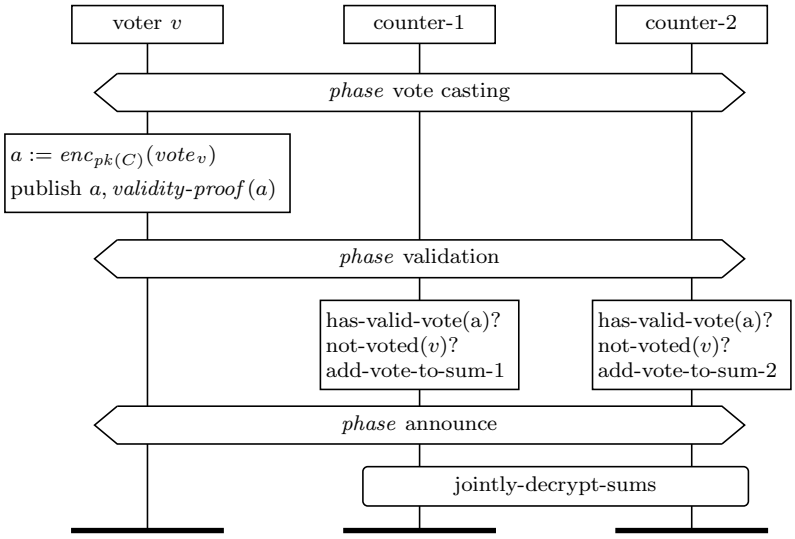


Figure 10: The CGS97 system with two counters.

Hirt and Sako [HS00] pointed out that in the specific multi-party example cited, participants can prove specific behaviour.

The CGS97 system reduced the communication and computational load for voting significantly, in comparison to previous efforts. In addition, the ideas set forth by this system have been influential, leading to various adaptations to achieve receipt-freeness.

Claims. CGS97 claims universal verifiability (UV) and and ballot-privacy (BP). In addition, it claims the following other security claims: robustness and prevention of vote duplication.

5.7. HS00: homomorphic tallying for randomised candidate lists

Hirt and Sako [HS00] set out to further reduce overhead and enable less strict physical assumptions. They cast a critical eye on previous work, showing a receipt in the multi-authority version of BT94, noting the processing load of SK95 and identifying the use of anonymous, secret communication channels in Oka97 as prohibitive towards adaption.

Hirt and Sako proposed a generic construction for introducing receipt-freeness in voting systems based on homomorphic cryptography. It reduces the strong assumptions from earlier works, but still requires an untappable channel from voting authorities to voters. Hirt and Sako proposed to randomise the candidate list, like BT94 and SK95. The HS00 system uses a mixnet to construct a randomised list of candidates. A zero knowledge proof of correctness of the mix is published, and a designated verifier proof revealing how the candidate list was mixed is given to the voter. The voter derives which entry in a shuffled, encrypted list represents her choice. This process greatly resembles vote casting in SK95 (Figure 15). The only exception is the last message. Instead of sending her vote over an anonymous channel, the voter announces it publicly. The counters take all published votes, homomorphically sum them together and jointly

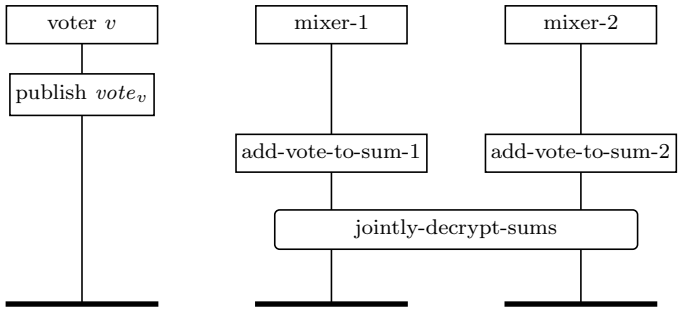


Figure 11: The HS00 vote casting phase (improving upon SK95).

decrypt the result, without reconstructing the decryption key. The vote casting phase is depicted in Figure 11. Note that all communications occur via the bulletin board.

Like [SK95], the used designated verifier proofs require possession of a private key. Hirt and Sako provided a protocol in which a verifier proves she possesses the private key. This ensures non-transferability of the proof.

Hirt and Sako applied their construction to CGS97. The lack of receipt-freeness in CGS97 originates from the fact that the voter creates and posts her vote on a public channel. This derivative consists of the basic construction by Hirt and Sako, and borrows the cryptographic scheme and the encoding of votes from CGS97. Consequently, the resulting system more closely resembles SK95 (as depicted in Figure 15) than CGS97 (as shown in Figure 10).

Claims. HS00 claims receipt-freeness (RF) and a threshold version of ballot privacy (any group with less than t colluding authorities cannot break privacy), and individual and universal verifiability (IV, UV). In addition, it claims correctness of tally. The authors also claim that as long as the voter knows of a non-colluding authority, the voter is even free from coercion.

5.8. LK00: third-party re-encryption of votes

Lee and Kim [LK00] also introduced receipt-freeness in CGS97. Unlike HS00, Lee and Kim did not aim to provide a generic construction for receipt-freeness, but seek to incorporate receipt-freeness into CGS97. Their approach is to ensure that the receipt present in CGS97 (the voter's vote) can no longer act as a receipt. They introduced a trusted third party, an honest verifier, that cooperates with the voter to construct her encrypted vote. Since neither party is fully aware of the randomness used to encrypt the vote, neither can prove the link between ciphertext and plaintext. For homomorphic tallying, it is necessary to prove that the resulting ciphertext is an encryption of a valid candidate (cf. Section 3.1). To prove validity of the encryption, voter and honest verifier must cooperate.

The vote encryption process, sketched in Figure 12, involves two partly overlapping challenge-response protocols: the first for the voter's encrypted vote $enc_{pk(C),n_v}(vote_v)$, the second for the verifier's randomness n_{vf}. Communication occurs via a public bulletin board.

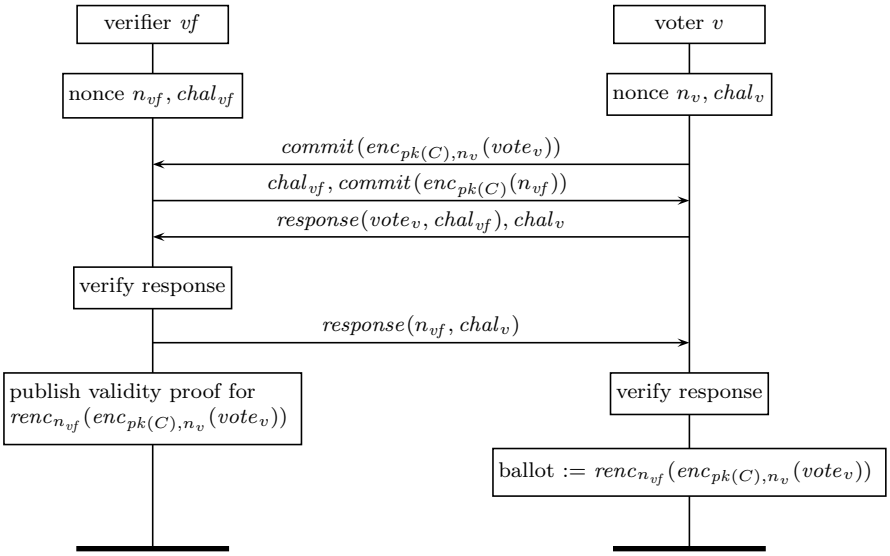


Figure 12: Verifier and voter jointly computing an encrypted ballot in LK00.

First, the voter publicly commits to her encrypted vote. In Figure 12 this concerns the message $commit(enc_{pk(C),n_v}(vote_v))$. The verifier responds with a challenge $chal_{vf}$ to the encrypted vote, and a commitment to her encrypted randomness n_{vf}. The voter computes the appropriate response $response(vote_v, chal_{vf})$ for the received challenge of her encrypted vote. She then sends the computed response and a challenge of the verifier's randomness, $chal_v$. If the response is correct, the verifier computes the correct response $response(n_{vf}, chal_v)$ to the voter's challenge, and computes and publishes a proof of validity for the re-encryption of the voter's encrypted vote. The voter re-encrypts her vote using the encrypted randomness $enc_{pk(C)}(n_{vf})$ of the verifier, and posts this on the bulletin board.

Claims. LK00 claims receipt-freeness (RF) and universal verifiability (UV). Additionally, it claims accuracy, eligibility, fairness, and robustness.

Studies. Hirt [Hir01] pointed out that the verifier can help the voter in casting a vote that falsifies the result. Furthermore, the voter can construct a receipt similarly to the receipt in the multi-authority version of BT94.

5.9. BFP01: improving scalability by distributed tallying

Very few systems in literature mention scalability of their approach. Moreover, scalability is usually a problem for actual elections. Operations such as mixing all votes together become prohibitively time consuming for constitutions with tens of millions of voters (such as the UK or France).

Baudron et al. [BFP+01] set out to address the scalability issue. The authors aim to design a voting system sufficiently scaleable to be used in large-scale elections, while still satisfying privacy and verifiability requirements. To this end, the proposed system

distributes the computation of the result over several aggregation levels. Their system first computes results at the local level, and then aggregates these at the regional level. Finally, the regional results are aggregated at the national level. The system achieves this by having the voter cast three encrypted votes (one for each aggregation level) along with a proof that the three encrypted votes encrypt the same candidate. To facilitate this, the authors introduced an encoding scheme nowadays known as Baudron counters (cf. Section 4.4).

As these votes are made public, the basic system is inherently not receipt-free. Baudron et al. addressed this by suggesting to use a hardware device to re-encrypt the votes (reminiscent of LK00). Correctness of the encryption can be proven using an interactive zero knowledge proof or a non-interactive, designated verifier proof. They claimed that the interactive zero knowledge proof would not harm receipt-freeness as the actual transcript of the interaction is indistinguishable from a simulated transcript (and thus the voter can provide a fake transcript). However, as pointed out by Hirt [Hir01] and later by Juels et al. [JCJ05], the interactivity in a zero knowledge protocol can be abused to construct a receipt. Moreover, the remark in [HS00] concerning designated verifier proofs is also not addressed. As Baudron et al. did not provide further details, we consider the claim of receipt-freeness for their protocol dubious at best.

Claims. BFP01 claims receipt-freeness (RF) and universal verifiability (UV). In addition it claims robustness against faulty authorities.

5.10. MBC01: smart cards re-encrypting votes

Magkos, Burmester and Chrissikopoulis again set out to transform CGS97 into a receipt-free voting system. Independent of Lee and Kim [LK00], they followed the same approach of trying to ensure that the obvious receipt of CGS97 is not a proof of how the voter voted. As in [LK00], randomness from a third party is used to encrypt the voter's vote. However, instead of relying on an external party, they implemented their system using a tamper-resistant smart card (see Figure 13). The voter sends a random order of the two possible votes (+1 and -1) to the smart card. The smart card then re-encrypts these values (denoted by *renc* in the figure) and sends them back. Finally, the smart card proves the correctness of the re-encryption using an interactive zero knowledge proof (denoted by the dotted arrow in Figure 13). After this, the voter knows which of a', b' is an encryption of her choice. After this, the system mimics the steps of the CGS97 system.

Magkos et al. identified a need for randomness in voting systems, to ensure vote-privacy. They analysed where this randomness could originate from: voter, authorities or an external source. According to their analysis, letting the voter introduce randomness enables her to abuse the randomness as a receipt. Letting the authorities choose the randomness implies a need for an untappable channel, according to this analysis. An external source would also require such a channel. However, Magkos et al. proposed to provide the voter with tamper-resistant smart cards for introducing randomness. This would reduce the assumptions further: as long as the voter can interact with the smart card in an unobserved, untappable way, receipt-freeness can be achieved.

The approach of Magkos et al. somewhat relaxes the constraints on communication channels, but leaves open how to ensure that the voter is not observed while using these

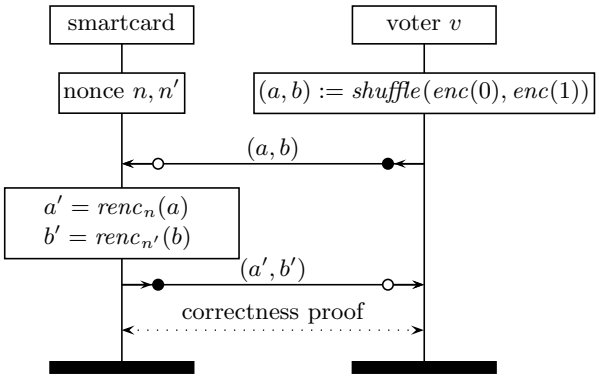


Figure 13: MBC01 extensions to ensure receipt-freeness in CGS97.

channels. Furthermore, their relaxation is achieved by introduction of a smart card. They did not answer the question whether this introduction brings new risks.

Claims. MBC01 claims receipt-freeness (RF) if using a tamper-free smart card, and makes no explicit verifiability claims.

Studies. Independent of each other, [LK02] and [JCJ05] identified the same method (using an intruder-supplied challenge) to arrive at a receipt in the MBC01 system, irrespective of tamper-freeness of the smart card.

5.11. Lee et al. series: Externally re-encrypted votes

In collaboration with others, Lee explored external re-encryption of votes to repair the LK00 system in three closely related systems. The first system uses dedicated hardware in a homomorphic tallying scheme (LK02), the second (LBD03) adopts this to a mixnet setting instead. The third system (ALBD04) tweaks the mixnet setting for the use of an optimistic mixnet system.

LK02: Smart cards re-encrypting votes. Lee and Kim heeded Hirt's remarks [Hir01] and noted that the approach to constructing a receipt for LK00 is also applicable to MBC01, as both voting systems use interactive proofs, where the voter may abuse her interactivity to construct a receipt. They also claimed that this holds true in general for blind signatures: the used blinding factor can act as a receipt.

Lee and Kim based their system [LK02] on the system proposed in [Hir01], but they replaced the assumption of a trusted third party randomiser therein with an assumption of a tamper-resistant randomiser possessed by the voter. The resulting system greatly resembles the MBC01 system (and is thus not depicted graphically), where the interactive proof is replaced with a designated verifier proof.

Lee and Kim noted that the costs of external hardware for every voter may be prohibitive to large-scale adoption. Although Lee and Kim did not state this explicitly, their system also relies on the assumption of designated verifier proofs, as pointed out in HS00 (i.e., designated verifier proofs are transferable to anyone who can be convinced

that the voter does not possess the decryption key). Furthermore, as Lee and Kim themselves pointed out, one of the proof-techniques they use may leak more information than intended.

In addition to the above remarks, the conclusions for MBC01 carry over to LK00: the constraints are relaxed by introducing a new object, but the security of this new object is not extensively treated.

LBD03: Mixing instead of homomorphic tallying. Lee et al. took the LK02 system and used mixnets instead of homomorphic tallying [LBD+04]. LBD03 does not propose any particular mixnet. Since LBD03 decrypts the individual votes, a proof of validity of the vote is no longer necessary, and thus dropped. Lee et al. noted that the malleability of the cryptographic system used in LBD03 enables a voter to copy another voter's encrypted vote. They seemed to imply that this does not pose a security risk, even though the signed and encrypted votes of each voter is cast by making it public.

ALBD04: Optimistic mixing. Aditya joined Lee et al. in [ALBD04], which proposes an optimistic mixnet for LBD03. In addition, the tamper-resistant hardware is removed by letting an authority fulfill the vote re-encryption tasks. The mixnet used is optimistic in that it only proves that the product of the input is the product of the output, not that there exists a strict correspondence between in- and output. Thus, a dishonest mix could mix votes c_1, c_2 as $\frac{1}{2}c_1$ and $2\ c_2$, since $\frac{1}{2}c_1 \cdot 2\ c_2 = c_1 \cdot c_2$. To deal with such situations, the system proposes a trace-back protocol which traces a vote back through the mixnet (by having each mix in turn undo its mixing for that vote) to find the faulty mix, in case the decrypted vote was invalid.

As Aditya et al. pointed out, the security of optimistic mixnets is not beyond doubt yet (see e.g. [Wik03]). They introduced several changes to the used optimistic mixing scheme to prevent identified flaws. Most notable of these changes are the use of only one encryption layer, and omitting the hashing. This latter change reintroduces the possibility to change votes as explained above.

Claims.

- LK02: receipt-freeness (RF) and universal verifiability (UV).
  Further claims: soundness, completeness, unreusability, eligibility, and fairness.
- LBD03: receipt-freeness (RF) and universal verifiability (UV).
  Further claims: prevention of double voting and fairness.
- ALBD04: receipt-freeness (RF), individual and universal verifiability (IV, UV).
  Further claims: eligibility, prevention of double voting, fairness and robustness.

Studies. Privacy of LBD03 is studied in a.o. [KT09, DKR09, DLL11].

5.12. JCJ05: introducing coercion-resistance

Juels, Catalano and Jakobsson [JCJ05] noticed that there remain privacy attacks for voters even if a system satisfies receipt-freeness. In particular, they identified forced abstention (the voter is forced to abstain), simulation (the voter is forced to give her credentials to the attacker who votes in her stead) and randomised voting (the voter

is forced to vote for a random candidate). They proposed a new notion of privacy, *coercion-resistance*, to capture the requirement of being receipt-free and preventing the three attacks, and propose a system to satisfy this requirement. This seminal system has laid the basis for the development of other election systems including the recently proposed Civitas [CCM08] (cf. Section 5.14).

1. $\mathcal{L}$:= "list of all published encrypted votes + credentials + proofs." $\mathcal{B}$:= "list of all valid, encrypted credentials published by the registrar."
2. Remove all rows with invalid proofs from $\mathcal{L}$.
3. Remove all rows with duplicate credentials from $\mathcal{L}$. E.g. by keeping the most recently cast row / vote.
4. Mix $\mathcal{L}$.
5. Remove fake credentials from $\mathcal{L}$. Achieved by plaintext-equivalent-tests between $\mathcal{B}$ and $\mathcal{L}$.
6. Decrypt remaining ballots on $\mathcal{L}$.
7. Publish $\mathcal{L}$.

Table 1: The counting process in JCJ05.

Juels, Catalano and Jakobsson's protocol uses zero-knowledge proofs and an anonymous public channel. Moreover, it assumes the channel between registrar and voters to be untappable. Voters receive a credential from the registrar in the registration phase. This credential constitutes a proof of eligibility and is used in the voting phase. Additionally, the registrar publishes the credential encrypted by the tallying authority's public key. In the voting phase, voters cast their vote on the anonymous channel by sending on the channel their vote, their encrypted credential and a zero-knowledge proof of the credential and of validity of the vote. In the end, the tallying authorities first check the proofs, then eliminate duplicates, next mix the ballots, then check the credentials and finally publish the set of valid votes. A plain-text equivalence test between the encrypted credential received from the voter and the published encrypted credential allows the tallying authorities to check the validity of the credentials without decrypting them, thus preserving their secrecy. The main reason why their protocol is coercion-resistant lies in the fact that a voter can derive fake credentials from the real one and use these fake credentials as the attacker pleases. Votes cast using these fake credentials are removed in the tallying phase. Since validity of credentials are only checked after the votes have been mixed (cf. Table 1: mixing precedes removal of fake credentials), the link between a removed vote and a published vote is not retrievable. The computational workload in JCJ05 is heavy, and several works [Smi05a, WAB07, AFT08, SKHS11] have suggested adaptations to address this issue.

Claims. JCJ05 claims receipt-freenesss (RF), coercion-resistance (CR), and universal verifiability (UV). Note that their definition of coercion-resistance implies receipt-freeness.

Studies. Privacy and correctness of JCJ05 are studied a.o. in [BHM08, DLL12].

5.13. MN06, MN07: Achieving unconditional privacy

Moran and Naor [MN06] argued the need for information-theoretic (or unconditional) privacy, as opposed to computational privacy achieved by encrypting votes. Moran and Naor remarked that regular encryption is computationally hiding (a computationally unbounded adversary can decrypt), but unconditionally binding: there is only one correct decryption. The Pedersen commitment scheme [Ped91] has the converse property: commitments are computationally binding, but unconditionally hiding. With this in mind, Moran and Naor proposed to use commitments instead of encryption. This recalls a prior system by Cramer et al. [CFSY96], which also uses commitments. However, this earlier system is not receipt-free.

To ensure privacy and verifiability of the commitments, Moran and Naor adopted the MarkPledge method (cf. Section 6.5). To ensure verifiability of the tally, they employed a cut-and-choose zero knowledge proof: the counter produces k new lists of commitments, each list being a masked permutation of the list of all voters' commitments (list 0). Then, a public random source outputs bits $1, \ldots, k$. If bit $i = 0$, list i is opened (proving the result of list i is the announced result). If bit $i = 1$, the permutation is revealed, proving that list i is a reordering of list 0.

This system relies on a trusted authority, who learns how each voter votes. In [MN07], Morat and Naor addressed this by introducing a way to secret-share commitments (adapted from the layered ballots of Punchscan). This removes the need for the MarkPledge method. Simply put, the commitment is shared between two authorities. They privately communicate about the randomness each used for the ballot order, thus enabling them to reconstruct the voter's choice.

Claims. Both MN06 and MN07 claim receipt-freeness (RF) and universal verifiability (UV). In addition, both claim "everlasting privacy" – the privacy of the link between voter and the choice they input does not depend on any computational assumption.

5.14. Civitas

Clarkson, Chong and Myers designed and implemented a remote voting system called Civitas [CCM08], which is considered as the *first* implemented system that is coercion-resistant and verifiable. The strong security provided by Civitas comes at the expense of its time-consuming tabulation phase. Civitas is based on JCJ05, but differs in several aspects. For instance, instead of assuming a single registration authority, Civitas has a registrar and a set of mutually distrusting registration tellers and introduces a construction of credential shares. The JCJ05 system does not describe how to distribute credentials, while a concrete protocol for this purpose is introduced in Civitas and is proven to be secure. In addition, Civitas provides concrete instantiations of cryptographic components which are left unspecified in JCJ05.

The security proof of Civitas extends the proof of JCJ05 to account for the changes discussed above. More importantly, an implementation of Civitas is made in a security-typed language, which enforces (information-flow) security policies. Based on experimental results, the authors claim that cost, tabulation time, and security of Civitas can be practical for real elections. As such, they claim that Civitas provides evidence that secure electronic voting can be made possible.

Claims. Civitas claims individual and universal verifiability (IV, UV) and coercion-resistance (CR).

Studies. Privacy of Civitas has been studied in a.o. [KT09].

5.15. Summary

The claims made by the various systems with respect to privacy and verifiability are listed in Table 2. Each row lists privacy claims made, verifiability claims made, known attacks, the main cryptographic technique used, and finally, works that investigated the system.

	privacy	*verifiability*	*attacks*	*based on*
FOO92	BP	UV	[KR05]⋆	Blind Sig
CGS97	BP	UV		Hom Enc
BT94	RF	–	[HS00]	Hom Enc
NR94	RF	–		Multiparty Comp
SK95	RF	UV	[MH96]	Mixnet
Oka96	RF	–	[Oka97]	Blind Sig, Commit
Oka97	RF	IV, UV		Blind Sig, Commit
HS00	RF	IV, UV		Mixnet
LK00	RF	UV	[LBD+04]	Hom Enc
BFP01	RF	UV	[Hir01, JCJ05]	Hom Enc
MBC01	RF	–	[LK02, JCJ05]	Hom Enc
LK02	RF	UV		Hom Enc
LBD03	RF	UV		Mixnet
ALBD04	RF	IV, UV		Mixnet
JCJ05	RF, CR	UV		Mixnet
MN06	everlasting BP, RF	UV		Commit
MN07	everlasting BP, RF	UV		Commit, Hom Enc

⋆ The problems found are due to unclarities, not to a fundamental problem with the system.

Table 2: Claims, subclassification, analyses of voting systems.

Privacy claims are listed as ballot-priv (ballot privacy), RF (receipt-freeness), CR (coercion-resistance), and everlasting BP (everlasting ballot privacy). Verifiability claims are listed as IV and UV, denoting individual and universal verifiability. Since voting systems are commonly categorised according to the main technique used to achieve privacy – traditionally: homomorphic encryption, blind signature, or mixnet – the table mentions this for each system. Note that NR94 is based on multiparty computations, and that MN06 and MN07 are based on commitments.

6. Systems focusing on verifiability

This section discusses notable practical systems proposed in literature. The main focus of this section is on end-to-end verifiable systems. As is common in research, ideas from one system influenced another. In Figure 14, we trace the stronger of such influences – in particular, we highlight where the core ideas that influenced a particular system originated. The upper part of the figure displays the systems that were directly or indirectly influenced by Chaum's Visual crypto, while at the bottom the more or less independently developed systems have been listed.

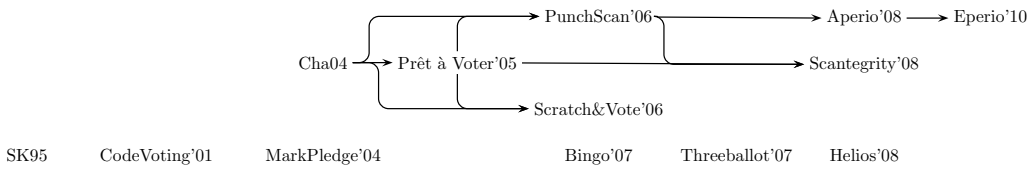


Figure 14: Influences between voting systems focusing on verifiability.

6.1. Overview

In this section, we turn our attention to systems that focus on improving practical verifiability. We begin with the work due to Sako and Killian, which defined the concept of *universal verifiability*, harmonizing previously existing verifiability concepts and terminology. Next, we describe the initial systems that further refined verifiability: Code Voting (introducing *recorded-as-cast* and *tallied-as-recorded*), Chaum's visual crypto voting system and Neff's MarkPledge(both of which introduced *cast-as-intended*). We then continue with systems inspired by these new verifiability concepts: Ryan's Prêt à Voter, which inspired Punchscan. We next discuss the systems Scratch & Vote, Threeballot, Bingo Voting and the Punchscan-inspired Scantegrity. Finally, we discuss Aperio/Eperio, and Helios. Whereas the previous section surveyed systems innovating on privacy aspects, this section surveys systems primarily focused on innovating verifiability aspects. Consequently, we see the claims of verifiability becoming more refined, going from individual/universal verifiability (IV, UV) to cast-as-intended (CAI), recorded-as-cast (RAC), and tallied-as-recorded (TAR).

6.2. SK95: introducing universal verifiability

Sako and Kilian [SK95] noticed the reliance on strong physical assumptions (voting booths) in both [BT94] and [NR94]. They proposed to use mixnets to relax the strong assumptions. Moreover, their mixnet-based approach is one of the first to use proofs of correctness for mixnets to achieve universal verifiability. Similar to [BT94], their system lets the voting authorities mix an encrypted 0 and an encrypted 1 vote, while proving to the voter how they mixed. This process is detailed in Figure 15 for an election with two mixers.

The last mixer generates a list of pairs (in Figure 15, the list contains one pair) of an encrypted zero and an encrypted one. The plaintexts are encrypted using a probabilistic encryption scheme, which ensures that no two ciphertexts are equal. The mixer publishes the list of pairs, and a proof that each pair contains an encrypted zero and an encrypted

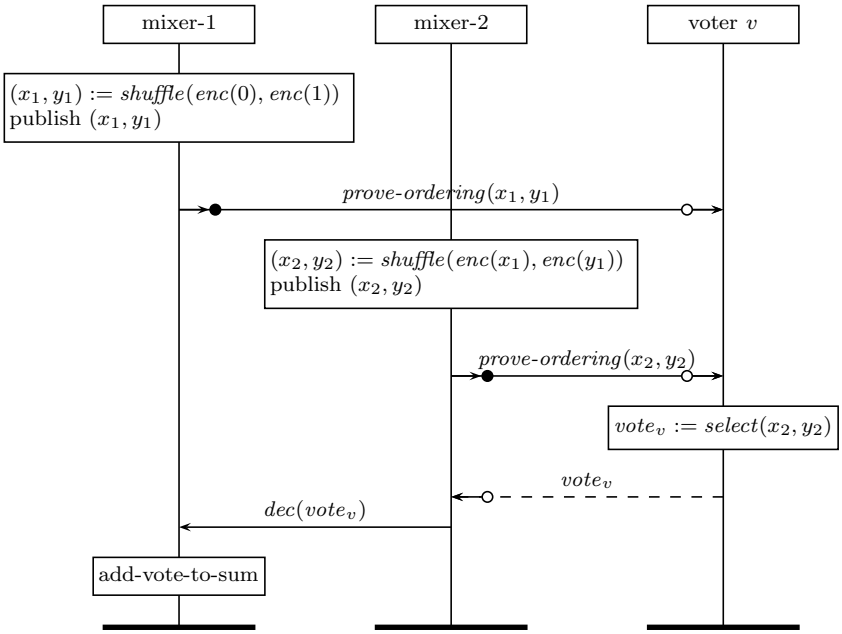


Figure 15: The SK95 system with two mixers.

one. Furthermore, the mixer sends a designated-verified proof to the voter of the order of each pair. This proof, sent over an untappable channel, is such that the voter can lie about the order. After this, the next mixer shuffles the list, after which he publishes the shuffled list and a proof that he shuffled correctly. Moreover, he sends a proof of how he reordered over an untappable channel to the voter. Again, this proof can be used by the voter to lie. This process is repeated by all mixers in the mixnet. After the last mix, the voter sends one element of one pair of the final list (an element corresponding to her choice) over an anonymous channel to the counter.

Sako and Kilian managed to relax the strict assumptions of previous systems (which require a voting booth). Instead, the system requires an anonymous channel and a private channel. However, their approach does not scale well to multi-candidate elections and their cryptographic approach does not offer full security. The system has been used as basis for an implementation, DigiShuff [FMM$^+$02]. DigiShuff mitigates the critique to a certain extent, which is discussed later.

Attack on the mixnet. Michels and Horster [MH96] described a known attack whereby a malicious voter can relate her vote to an honest voter's vote, by merely using the encrypted version of the honest vote. This relation survives decryption. Thus, the dishonest voter can later test all published votes to see which two satisfy the relation, thereby revealing (to the malicious voter) how the honest voter voted. In addition to this attack, Michels and Horster also described a new attack against the mixing process. The shuffle of a mix j can be undone if all subsequent mixes $j+1, \ldots, n$ collaborate. Note that this means that if there is only one honest mix (which should be enough to provide an honest mixnet), its shuffling can be completely negated by the rest of the mixnet –

without knowing how it shuffled.

Claims. SK95 claims receipt-freeness (RF) and universal verifiability (UV).

Studies. Michels and Horster [MH96] identified a flaw in the mixing scheme. The DigiShuff [FMM+02] system is an implementation based on SK95.

Candidate	*Vote code*	*Acknowledgment code*
North	5432	314159
East	1234	271828
South	7890	141421
West	7654	161803

Figure 16: A code sheet for Code Voting.

6.3. Code Voting & Pretty Good Democracy

Code voting is originally due to Chaum [Cha01]. The central idea is to protect a voter against a malicious computer. To achieve this, the voter receives a code sheet (cf. Figure 16) via non-digital means. The codes on the sheet are random. To vote for a candidate, the voter enters the *Vote code*. Since the codes are random, even a malicious computer cannot change this in a meaningful way. The voter knows her vote was received, when her computer shows the correct acknowledgment code. This enables a form of cast-as-intended.

Since Code Voting's inception, this central idea has been investigated by others, e.g. [JRF10, HS07, HRT10]. The central idea has also been further explored for online voting systems, such as Pretty Good Democracy (PGD [HRT10]). PGD extends the code voting guarantees to also give some assurance to the voter that her vote was properly recorded. This is done by storing all codes (including the acknowledgment code) encrypted at the server side. Therefore, receiving a correct ack code ensures that at least a threshold set of authorities agreed on the received vote code.

Claims. Code Voting claims cast-as-intended (CAI), recorded-as-cast (RAC), and ballot-privacy (BP). PGD claims CAI, RAC (if a threshold set of authorities is honest), tallied-as-cast(TAR), and receipt-freeness (RF). The authors point out that PGD fails to satisfy coercion-resistance (no CR).

6.4. Chaum04: Visual crypto ensuring correctness of encryption

In [Cha04], Chaum identified that verifiability as offered by voting systems was relying far too much on cryptography to be of practical use to voters. He applied visual cryptography to voting. Visual crypto is the trick of separating an image into multiple shares. By themselves, each share contains random information. Only when all shares are overlayed, does the original image reappear. The construction of the shares is explained in Figure 17.

Every pixel of the original image is translated into a square with four sub-pixels. The shares are stacked. Therefore, a white pixel results in two identical squares for the two shares (either ▚ + ▚, or ▞ + ▞), while a black pixel results in two complementary squares

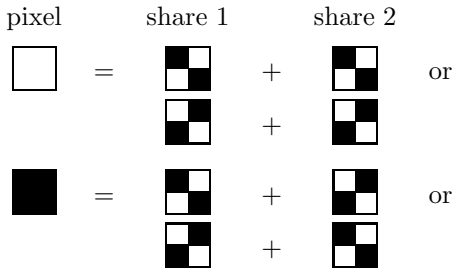


Figure 17: Visual crypto: the construction of shares.

(either ■ + ■, or ■ + ■). Randomness of the shares is achieved by randomly selecting for each pixel either the first configuration or the second configuration. The selection can, e.g., be based on a cryptographic key. Chaum leveraged this to combine easy voter verification with vote-privacy. A voter's vote is recorded on two layers of paper. Each layer contains a pattern of apparently random dots, based on the mixnet's public key. When viewing one layer over the other, the voter's vote is revealed (see Figure 18). The voter can take either one of the two layers as a receipt, as neither layer by itself reveals her vote – they only show random dots (except by someone who possesses the decryption "key" (layer)). She selects one layer to take home, and the other layer is destroyed. The kept layer is scanned and acts as the voter's ballot.

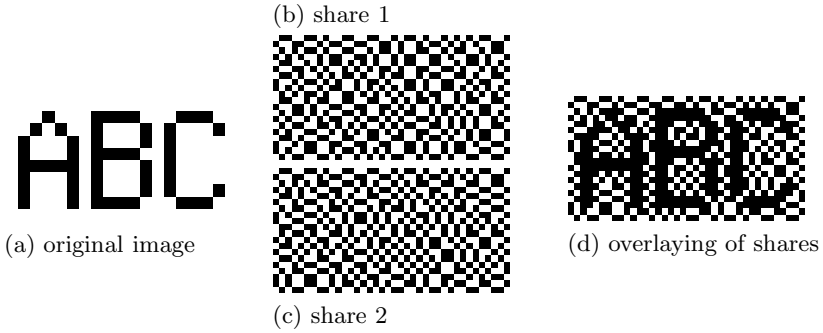


Figure 18: Visual crypto: splitting an image into two shares and recombining the shares.

All scanned ballots are published, enabling[1] the voter to verify that her ballot is received correctly. To determine the election result from the ballots, the ballots are mixed. Each mix visually decrypts the ballots, analogous to a decryption mix (in effect, adding a layer containing their share of the key to the ballot). This results in a set of readable ballots, from which the tally is straightforwardly established. To ensure verifiability, Chaum suggested to use randomised partial checking [JJR02] (detailed in the description of Scantegrity below).

Chaum's approach to combining privacy and verifiability does not require heavy cryptographic operations on the part of the voter. However, the mixing process, which uses

[1]Chaum does not detail how the voter can pick out her ballot from amongst all ballots.

a shared visual cryptographic key, is still a complex process. It is not clear whether this process is understandable for an average voter. As verifiability of the mixing process is necessary to achieve verifiability, it is not clear whether the goal of practical and simple verification has been achieved.

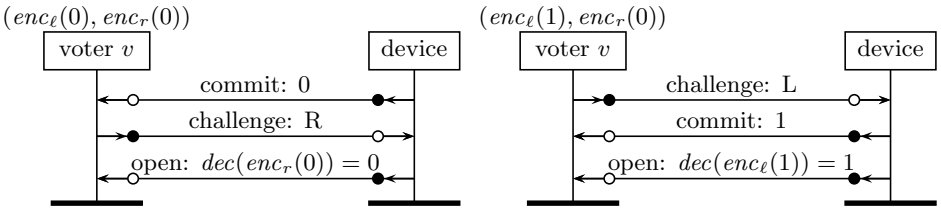
Claims. This system claims cast-as-intended (CAI, noting CAI fails if the system can predict which of the two layers the voter will retain), recorded-as-cast(RAC, noting RAC fails if the system predicts two voters that choose to retain the same layer), tallied-as-recorded(CAR), and receipt-freeness (RF).

Studies. This system has been studied in a.o. [PV08].

6.5. MarkPledge: verifying ballot encryption

Independently, Neff also proposed a way to achieve practical verifiability. The MarkPledge system [Nef04] aims to ensure to the voter that the encryption made by a voting device is indeed a correct encryption of her choice. This is achieved by means of a challenge-response protocol as follows.

The vote (yes/no) for each candidate is encoded as a list of pairs of encrypted bits $(enc_r(b), enc_{r'}(b'))$. When $b = b'$ (for every pair in the list), the vote encodes a "yes", and when $b \neq b'$ for every pair in the list, it encodes a "no". A commitment consists of a bitstring, e.g. $0, 1, 1, 1, 0, \ldots$. A challenge simply states whether to open (prove the decryption of) the left or the right encrypted bit for every pair. All commitments, challenges and responses are printed on one line with the candidate name, and the security of the verification method is determined by the length of the list. More precisely, for a list of length n, the chance of the opening being correct while the list differs is 2^{-n} [JR12].



(a) verifying correct encryption (b) preserving privacy

Figure 19: Verification and privacy of one bit-encryption in MarkPledge.

Verification that the vote is for the chosen candidate is achieved by challenging *after* committing. In Figure 19(a), this is shown for a yes-vote (i.e., a pair of ciphertexts encrypting the same value): the device first commits to the value encrypted – in this case, 0. Now the voter can challenge the device to decrypt either the left or the right ciphertext. After making her choice for right, the device proceeds to decrypt the right ciphertext, which matches the value the device committed to. As the device cannot predict what the voter will choose, the device has a 50% chance per encrypted pair of being detected if cheating. To ensure privacy, the voter supplies the device her challenges

before the device commits. This is shown in Figure 19(b). Given these challenges, the device can create fake proofs of "yes" votes for the other candidates

Therefore, the voter's printed receipt shows she voted "yes" for every candidate. Only if a challenge occurred before the commit is this a real proof. The voter knows this, but cannot prove it to anyone else (preserving privacy).

The voter only needs to verify that challenge and commit are printed on the same line as the candidate to which they belong. Any helper organisation can then verify correctness of the whole receipt, thus completing verification. Note that correctness is only checked for "yes" encryptions – complete ballot correctness can be checked by other means [AN06].

The challenge-response protocol occurs in a voting booth, ensuring privacy (avoiding Hirt's attack on BT94).

The verification technique of MarkPledge has been adopted by a number of systems, such as [JRF09] (combining MarkPledge with code voting), [MN06] (combining MarkPledge with a commitment scheme), and [JR12] (using a new encoding of the mechanism in terms of points on a line).

Claims. MarkPledge claims cast-as-intended (CAI), universal verifiability (UV), receipt-freeness (RF), and coercion-resistance (CR).

6.6. Prêt à Voter: randomised candidate orders

Prêt à Voter implements the concept of combining privacy and verifiability in a tangible way. An initial version was developed by Ryan [Rya05], which served as the basis for the system carrying the name Prêt à Voter due to Chaum, Ryan and Schneider [CRS05]. This system avoids the use of a two-layered receipt by having two columns (see Figure 20). The left-hand column of the ballot presents the candidates in a random order, while the right-hand column has space for the voter to mark her preference. At the bottom of the right-hand column is a cryptographic string (an onion) which encodes the candidate ordering on the ballot. This onion is the encryption of a random offset D_0 with each counter's public keys.

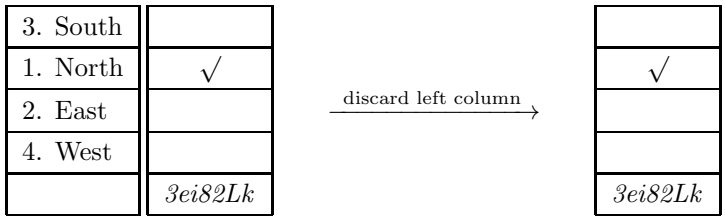


Figure 20: A Prêt à Voter ballot voting for North, where $dec(3ei82Lk) = (3, 1, 2, 4)$.

A voter marks her preference, separates the two columns and destroys the left-hand column. The right-hand column is scanned by the system and taken home by the voter as a receipt. Since the candidate ordering is not retrievable from the right-hand column except by all counters decrypting in order, the receipt cannot be used to prove the voter's preference.

Creating a candidate order is handled in a similar manner to HS00. The main difference is that Prêt à Voter does not prove the candidate order, but encrypts it on the

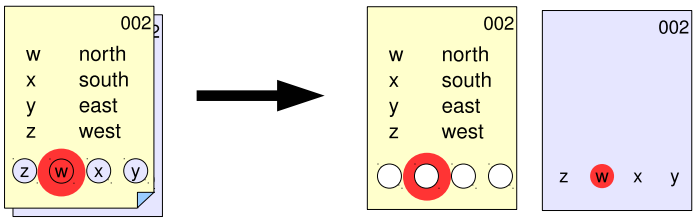


Figure 21: A *Punchscan* ballot (voting "north") consists of two layers.

ballot's right hand column. Correctness is ensured by a ballot auditing procedure, which decrypts the onion on a given ballot (thus showing whether or not the onion matches that ballot's candidate order). This voids the ballot in the process, hence Prêt à Voter requires generation of surplus ballots.

To provide further verifiability without sacrificing privacy, the conversion of receipts to countable votes is subject to randomised partial checking, cf. Section 3.7.

Various works have followed up on Prêt à Voter to address issues such as chain voting and kleptography, and to extend the capabilities of the system to support more diverse voting methods (such as single transferable vote), and to unify the new results into one coherent voting system. Due to these efforts, the Prêt à Voter system has matured and become more versatile. Development of Prêt à Voter has actively continued, resulting in user trials [SLC+11], and plans for use in actual elections (in Victoria, Australia).

Claims. Prêt à Voter claims cast-as-intended (CAI), recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF).

Studies. Prêt à Voter has been studied in a.o. [PV08, JMP09b, KRMC10].

6.7. Punchscan: layered paper ballots

Punchscan [FCS06] merges Chaum's idea of layers with Prêt à Voter's concept of randomised ordering. In Punchscan, a ballot consists of two layers (see Figure 21). The top layer has a numbered list of the candidates in a random order (as in Prêt à Voter). Below that list, there is a series of holes. The bottom layer provides the used numbers in a random order (again following Prêt à Voter) at the location of the holes in the top layer. A choice is made by using a big marker to colour both the bottom layer number and the top layer hole at the same time. The top layer does not identify which choice has been made, only which hole has been selected. The bottom layer does not reveal which candidate has been selected, only which sequence number that candidate had on this particular ballot. Hence, neither ballot can identify the voter's choice. One layer is destroyed, and the other one is copied for tallying. The voter retains the copied layer for verification.

The ordering of sequence numbers in both top and bottom layer is stored by the authorities, and linked to the ballot id number (in Figure 21: 002). Using the stored information, the authorities can reconstruct the voter's choice from either layer in a manner similar to Prêt à Voter.

One notable drawback of Punchscan, identified in [BMR07, MN07], is that even though a layer does not give sufficient information to determine how a voter voted, the

distribution of layers over possible votes can be skewed. For example, consider an election between two choices (Figure 22).

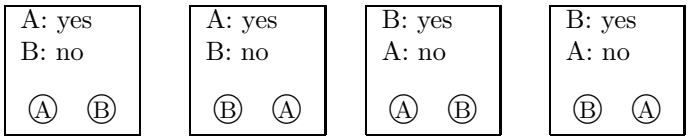


Figure 22: Ballot orderings for a two-choice election in Punchscan (adapted from [BMR07]).

An attacker may now offer a reward for:

- any top layer where Yes is labelled A, and where the left hole is marked,
- any bottom layer where A appears left and is marked.

All possible ballots are shown in Figure 22. Note that only for the first ballot, voting Yes will leave the result with a rewarding layer. To receive a reward if given the second or third ballot, a voter must vote No. And, if given the fourth ballot, the voter cannot produce a layer that will receive the reward. Thus, an obedient or greedy voter is likely to vote No.

This particular drawback was addressed in Punchscan by forcing a voter to choose which layer is her receipt before she has seen the ballot.

Claims. Punchscan claims recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF).

Studies. Punchscan is studied in a.o. [BMR07, PV08, KRMC10]. The similarities and differences with Prêt à Voter are studied in [vdG09]. The partial attack on Punchscan's receipt-freeness mentioned above is described in [BMR07, MN07]. In 2007, Punchscan was used for student elections [ECCP07] in Ottawa, Canada.

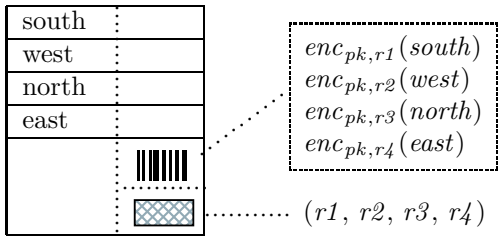


Figure 23: Example of a *Scratch & Vote* ballot.

6.8. Scratch & Vote: verification using scratch strips

Adida and Rivest used scratch strips to simplify the ballot auditing from Prêt à Voter. Their Scratch & Vote system [AR06] uses similar ballots as Prêt à Voter, but replaces the onion with a barcode and a scratchable surface (see Figure 23). The barcode contains an encryption of the candidate order, and below the scratch surface the randomness used in

the encryption is printed. The voter receives two ballots: one to audit and one to cast. Auditing consists of scratching away the scratch layer (which invalidates the ballot), and confirming correctness of the barcode (using a trusted helper). Casting works like in Prêt à Voter: the left-hand side is removed and discarded, and the right-hand side is handed to the election authority. He confirms that the scratch surface is untouched, detaches and destroys the scratch surface. This ensures that the right-hand side by itself cannot reveal the decrypted candidate order, thus preserving privacy.

Counting is performed by homomorphically adding all encrypted ballots, which uses Baudron-style counters (cf. Section 4.4). The encrypted result is then decrypted without revealing the decryption key (cf. Section 3.6).

Adida and Rivest extended the concept of scratch-surface verifiability to both Prêt à Voter and Punchscan. However, they noted that privacy of the system is based on the assumption that the scratch surface is perfect: either it hides the used random values, or it is detected as having been scratched. Furthermore, malicious authorities could record the candidate orderings in advance, and use this to reveal how voters voted. Adida and Rivest remarked that this problem is inherent in preprinted paper-based cryptographic voting. For example, Prêt à Voter and Punchscan have the same weakness.

Claims. Scratch & Vote claimes cast-as-intended (CAI), recorded-as-cast (RAC), tallied-as-recorded (TAR) and receipt-freeness (RF). The authors explicitly note that Scratch & Vote is susceptible to forced randomisation – as such, it does not satisfy coercion-resistance (no CR).

6.9. Threeballot: splitting a vote in three

A radically different approach to end-to-end verifiability was proposed by Rivest. His system, the Threeballot Scheme [RS07], aims to use no cryptography for the voter, while offering verifiability and privacy. To achieve this, he uses a specific form of ballot. This ballot consists of three separate ballots, which together form the whole ballot (a Threeballot, see Figure 24). Each separate ballot contains an election-wide unique code.

To vote, a voter marks each candidate once, and the candidate of her choice twice. She may place the marks on any of the ballots, as long as each row contains one mark, while her preferred candidate's row has two marks. When the voter casts the vote, she secretly chooses one ballot of which she receives a certified copy. She separates her Threeballot into 3 ballots, and casts all three.

After voting, the set of all received ballots is published. This allows all voters to verify that a ballot matching their certified copy is in the set. In addition, it also enables anyone to verify correctness of the announced result. The certified copy does not prove how the voter voted, as it could have been a part of other valid Threeballots.

However, as already noted in [RS07], not all ballots can be validly combined with a given receipt. For example, a receipt marking each candidate (the ballot on the right in Figure 24) can only be linked to one empty ballot and one ballot where precisely one candidate is marked. This lack of ubiquitous recombining of ballots enables an adversary to significantly reduce the privacy of a voter. The authors propose several solutions to address this. Each solution has its own problems with respect to feasibility and practicality.

Together with Smith, Rivest introduced two additional voting systems in [RS07]: VAV (vote/antivote/vote) and Twin. VAV greatly resembles Threeballot, but interprets

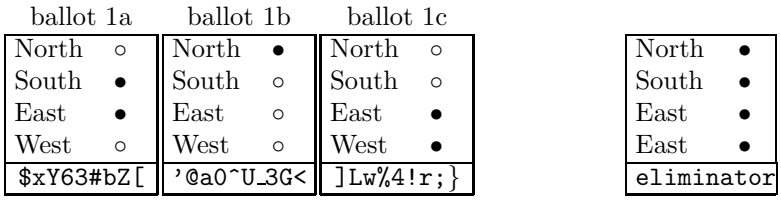


Figure 24: A *Threeballot* voting for "East" (left), and a hard-to-combine ballot (right).

the ballots differently. Two ballot are marked 'V' for vote, the third is marked 'A' for anti-vote. A mark on the anti-vote ballot cancels a mark on a vote ballot. The VAV system requires that the anti-vote ballot is marked precisely the same as one of the two vote ballots. Other than this, the system works as Threeballot (e.g. the voter can choose which ballot to copy as a receipt). In the third proposed system, Twin, each voter casts one ballot (not three), and takes home a receipt of *another* vote. Such receipts are called *floating receipts*. They trade direct verifiability off for increased privacy.

Rivest and Smith already pointed out several possible avenues of attack on Threeballot in their original paper. The level of privacy Threeballot offers has been the subject of criticism (see e.g., [App06]). Nevertheless, Threeballot's core ideas offer an appealing approach to combining verifiability and privacy without using cryptography.

Claims. Threeballot, VAV, and Twin all claim recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF). In addition, Threeballot and VAV claim cast-as-intended (CAI).

Studies. Privacy in Threeballot has been extensively studied, see e.g. [KTV10, App06, ACvdG10].

6.10. Bingo Voting

Bohli, Müller-Quade and Röhrich proposed to use a bingo machine as a source of randomness in their voting system [BMR07], hence the name Bingo Voting. Prior to the elections, the authorities generate for each voter, one random number per candidate. Then they publicly commit to the set of generated numbers. Thus, for n voters and m candidates, the authorities generate $n \cdot m$ random candidate-number pairs. The authorities keep these pairs secret, but publish commitments to these pairs.

To vote, a voter enters the voting booth and selects her candidate on a voting device. Once selected, a new random number is generated by a disconnected random number generator. Bohli et al. suggest to use (for example) a mechanical device such as used in bingo. The generated number is then transferred to the voting device and assigned to the chosen candidate. The system then generates a receipt, on which the candidate of her choice is assigned the freshly generated random number, and the other candidates are assigned numbers from the set of a priori generated numbers. In this way, the receipt lists one random number per candidate. The only difference between the chosen candidate's number and the other numbers is the freshness of the number.

In order to produce the election result, the list of all handed-out receipts is published. Furthermore, a list of all unused pairs from the prior set is published and opened, and

the authorities prove that all unopened commitments are indeed used on one receipt. The final result is then given by the unused, opened candidate-number pairs. Note that an abstaining voter leaves one unused candidate-number pair per candidate. Hence, the abstaining voters can be deducted from the tally, if necessary.

Bohli et al. claimed that Bingo Voting satisfies (amongst others) cast-as-intended, because a voter can verify that the freshly generated number is assigned to the candidate of her choice. However, during casting the voter has no opportunity to verify that the other numbers assigned are numbers that were committed to previously, nor that exactly one number was used from each set of candidate-number pairs of the non-preferred candidates.

Claims. Bingo claims cast-as-intended (CAI), recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF).

Studies. Coercion-resistance of Bingo voting is analysed in [KTV10, DLL11, DLL12].

6.11. Scantegrity: a voter-verifiable voting front-end

The researchers behind Punchscan recognised that the process of randomised partial checking and reconstructing the choice (originating from Prêt à Voter) can be taken independent from the ballot-casting process. They devised a new verification system called Scantegrity [CEC+08], relying on permutations of candidate IDs as in Punchscan, but using only one piece of paper. This verification can be added to an existing voting system to provide verifiability, and works as follows.

A ballot lists the candidates (e.g., Alice) and, for each candidate, an code (e.g., A). The assignment of codes to candidates is permuted for every ballot – on the next ballot A can be assigned to Bob. Voters mark a bubble next to the candidate of their choice, and can verify that their choice is correctly recorded by checking whether the pair (candidate code, ballot ID) appears correctly on the bulletin board. Moreover, to preserve privacy in computing the result, Scantegrity employs a "Switchboard" – a mix that mixes candidates (and their marks), instead of whole ballots (see Figure 25). Correctness of this process is proven using randomised partial checking, cf. Section 3.7.

Randomised partial checking uses two mixes to mix the votes, making the results of each mix public. Verifiability of the entire mixing process, depicted in Figure 25, is achieved by revealing for each vote in the intermediate mixed set either where it originated from (revealing the link to the input), or where it ended up (revealing the link to the output).

Both Scantegrity and Punchscan suffer from drawbacks. Neither prevents against the pattern voting attack (which motivated BT94). In this attack, a coerced voter marks her ballot with a unique pattern provided by the coercer, which allows the coercer to distinguish this ballot from all others. Moreover, neither prevents [CCC+08] against randomisation attacks (which motivated JCJ05). Furthermore, dispute resolution (when a voter claims her vote is inaccurately recorded) is complex in Scantegrity: a voter may claim her ballot is incorrectly recorded, but she could be lying. To preserve privacy while investigating this requires a two-stage process where first the ballot ID of the disputed ballot is matched (the rest of the ballot being obscured), and next the codes of a set containing dummy votes and the disputed ballot are revealed (with the ballot ID obscured).

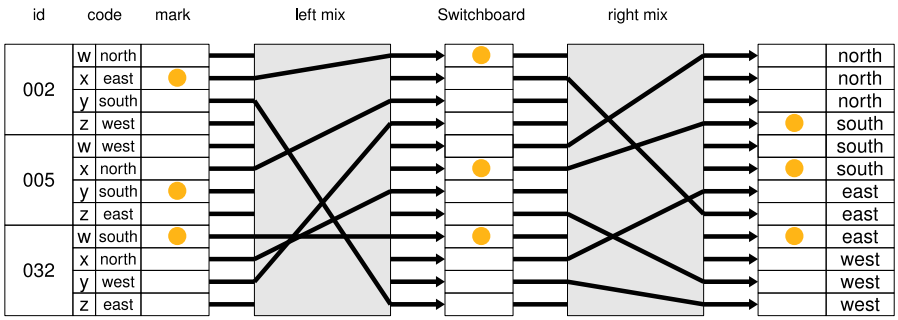


Figure 25: Randomised partial checking of *Scantegrity*'s candidate mixing.

In response to such issues and usability problems, Scantegrity II [CCC+08] has been developed. The main improvement is using random candidate codes, which are hidden by invisible ink. Using a special marker, the voter marks her choice, which reveals the candidate code printed invisibly in the bubble invisible ink. Thanks to this innovation, dispute resolution was greatly simplified. Since all genuine candidate codes are hidden by the invisible ink, a voter cannot guess a genuine code. Therefore, if she claims that her ballot is missing, and she has a valid candidate code not present, then her complaint is in good faith – the system is in the wrong. The paper addresses the risks introduced by using invisible ink, and the team provides a separate technical report on this via their website[2]. They conclude that it is feasible to use invisible ink. They do not discuss the increase in costs.

Claims. Both Scantegrity I and II claim recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF). Scantegrity II uses Prêt à Voter-style ballot auditing to achieve cast-as-intended (CAI).

Studies. Scantegrity is studied in [PV08]. The Scantegrity team is working to have Scantegrity adopted for elections, which has already resulted in Scantegrity II's use in a (local) election [CCC+10].

6.12. Aperio/Eperio

The Aperio system [ECA08] is a variation on Punchscan that was developed to serve as an end-to-end verifiable system for environments where computerised support is low (e.g., developing countries). Aperio extends the layered approach of Punchscan further, basically providing a layer-based approach to randomised partial auditing. Each ballot has one receipt layer and two or more (differently coloured) audit layers. Each layer has its own unique reference number for auditing (see Figure 26). As in Punchscan, the candidate order is randomised.

Prior to elections, two lists are generated for each audit layer: one linking the audit reference numbers to the receipt identifiers, and one linking the reference numbers to the candidate order. Figure 27 shows an example of these lists for the last audit layer.

[2] `http://scantegrity.org/~carback1/ink/ink.pdf`

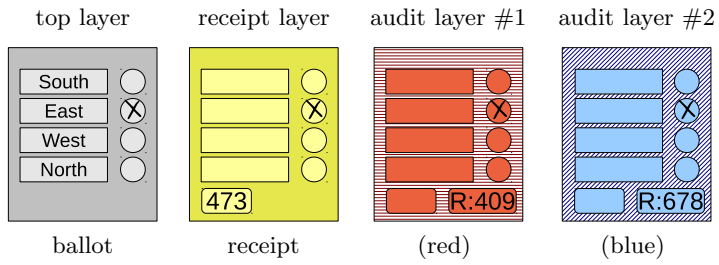


Figure 26: A 4-layered Aperio ballot voting for "east" (adapted from a presentation by Essex).

These lists are sealed into an appropriately-colored envelope, and the nature of the list (whether it links to receipts or the order) is noted on the envelope.

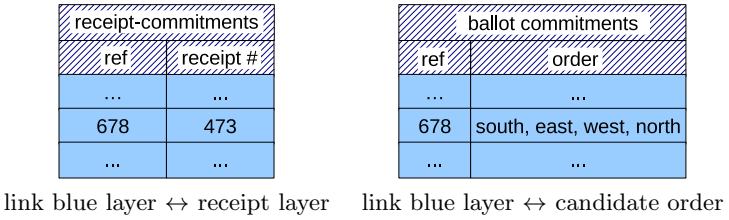


Figure 27: Audit lists for blue layer (adapted from a presentation by Essex).

To vote, a voter marks her choice on the top layer, which also marks the location of her choice on all underlying layers. The voter can take her receipt layer home. All other layers are cast into their appropriately-colored boxes – the ballot into the ballot box, and each audit layer into its own "audit box". Tallying is now straightforward by counting the votes from the ballot box.

Verifying correctness of the tally is achieved as follows: both envelopes of one color are taken, and the "receipt" envelope is destroyed without opening (to preserve privacy). The "order" envelope is opened, and used to recover the order for each audit layer of this color. By comparing the audit box result with the result of the ballot box, any ballot stuffing / deleting can be detected.

Similarly, tampering with ballots can be detected by using the "receipt" list to determine the correspondence between receipts and audit layers. This will detect any changes made to cast ballots.

Randomised partial auditing is achieved by choosing which colour of layer will reveal which of the two links. By using multiple audit layers, both the link to the voter ballot and the link to the result can be revealed without violating privacy. In this fashion, the system provides a level of verifiability.

The main concepts of Aperio were used in the fully digital Eperio [ECHA10] voting system. Fundamentally, Eperio works as Aperio does, but Eperio relies on cryptography (such as shared secrets, commitment schemes) to preserve privacy.

Claims. Both Aperio and Eperio claim cast-as-intended (CAI), recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF).

6.13. Helios

Adida designed an open-source, web-based voting system, Helios [Adi08], aimed at elections with low risk of coercion. As Adida put it, Helios claims "no novelty", but succeeds in taking existing ideas and implementing them in a working system. Voting in Helios uses a simplified version of Benaloh challenge [Ben07] (deemed acceptable for the low-coercion-risk environment) and the mixnet of [SK95].

To cast a vote, the voter enters her choice. The system then prepares an encrypted ballot that the voter can either audit or cast. Only when the voter chooses to cast her vote, she has to authenticate herself. Individual verifiability is achieved by posting the encrypted ballot to a public bulletin board. To tally, the votes are mixed and then decrypted and published on the bulletin board, as are all proofs of correctness.

Helios was updated and used for presidential elections of the Belgian UCL university [AdMPQ09]. The main improvements of this update were to replace mixing with homomorphic tallying from CGS97, and to add distributed decryption. Since then, Helios has been used in various other elections, including student elections at Princeton, an IACR test election and student elections at UCL. For that last election, the system moved to a new back end based on mixnets. Where previously, each candidate was encrypted, in this new version, only a ballot is encrypted, resulting in an impressive speed up.

Claims. Helios claims cast-as-intended (CAI), recorded-as-cast (RAC), tallied-as-recorded (TAR), and receipt-freeness (RF). The author explicitly notes that Helios only works in low-coercion environments – as such, it does not offer coercion-resistance (no CR).

Studies. Security of the Helios interface is analysed in [ED10]. Several related attacks on privacy, all leveraging the malleability of the used cryptographic scheme, are described [CS11]. These issues can be addressed by using a non-malleable cryptographic scheme [BCP+11].

6.14. Summary

Various efforts have emerged that sought to transform theoretical voting systems into actual systems. The designed systems aim to combine strong privacy measures with a high level of verifiability. However, these new approaches must be handled carefully – as evidenced by the privacy-reducing threats to Punchscan and Threeballot. Finally, we observe that researchers are reaching out to election officials and several systems (Punchscan, Scantegrity, Prêt à Voter, Helios) have been used in a variety of local elections, ranging from student elections, via university presidential elections, to municipal elections.

In Table 3, the findings of the end-to-end survey are summarised. In this table, BP refers to ballot privacy, and cai, rac, and tar refer to cast-as-intended, recorded-as-cast, and tallied-as-recorded, respectively. For each system, we list if it is intended for a remote (R) or supervised (S) environment, as well as the main techniques on which it is based.

7. Conclusion: trends and developments observed

In the above, we detailed the two major developments in the field of voting: first, a focus on privacy, which was later followed by a focus on practical verifiability. There

	priv	*verifiability*	*studies*	*based on*	⋆
Code voting	BP	RAC		codes	R
PGD	RF	CAI, RAC, TAR		Code Voting	R
Chaum04	RF	CAI	[PV08]	visual crypto	S
MarkPledge	CR	CAI, UV		challenge-response	R
Prêt à Voter (PaV)	RF	CAI, RAC, TAR	[PV08, JMP09b, KRMC10]	randomised order	S
Punchscan	RF	RAC, TAR	[BMR07, PV08, KRMC10]	PaV, layered ballot	S
Scantegrity	RF	RAC, TAR	[PV08]	Punchscan	S
Scantegrity II	RF	RAC, TAR		Scantegrity	S
Scratch & Vote	RF	CAI, RAC, TAR		PaV, scratch strips	S
ThreeBallot	RF	CAI, RAC, TAR	[KTV10, App06, ACvdG10]	3 ballots / voter	S
VAV	RF	CAI, RAC, TAR		ThreeBallot	S
Twin	RF	CAR, RAC		floating receipts	S
Bingo	RF	CAI, RAC, TAR	[KTV10, DLL11, DLL12]	PaV, MarkPledge	S
Aperio	RF	RAC, TAR		Punchscan	S
Aperio	RF	RAC, TAR		Aperio	S
Helios	RF	CAI, RAC, TAR	[ED10, CS11, BCP$^+$11]	CGS97	R
Civitas	CR	RAC, TAR		JCJ05	R

⋆ (R)emote or (S)upervised voting environment

Table 3: Properties of voting systems.

are various similarities between these developments: in the privacy line, refinements to the notion of privacy (receipt-freeness, coercion-resistance) were followed by individual attempts to satisfy the refined notion. More and more works claim to satisfy a notion, some of them pointing out problems in earlier work. The continued finding of problems in some earlier works underlines the need for a uniform approach to testing the validity of these claims. This leads to the appearance of studies of these claims. While these studies themselves were not the main subject of this paper, we do see a similar development there: At first, studies analysing a particular claim of an individual work appear [KR05, MVdV07, TMT$^+$08]. This is followed by the appearance of generic (formal) frameworks to study a claim of any work. Initially, these frameworks are theoretical [JdV06, BRS07, DKR09, JMP09b, JMP09a] and enable a uniform approach to validating claims. Later frameworks [BHM08, KT09], become more detailed, refine the notions under consideration and begin to consider tool support.

When the focus shifted towards incorporating verifiability, we see a similar trend: at first, a few works identify specific verifiability notions not yet suitably covered before, and propose individual solutions for these. Other works appear, some of which identify problems in earlier works. Recently, the first studying framework [KRS10] appeared, which enables manual proofs of individual, universal and eligibility verifiability.

If developments in verifiability continue similarly as developments in privacy did, then more verifiability frameworks will emerge, which will cover more refined notions such as

cast-as-intended, recorded-as-cast, and tallied-as-recorded. In addition, steps will be made towards automated verification.

Another trigger for new developments may arise from the design of new cryptographic building blocks. An example is the newly proposed fully homomorphic cryptosystem [Gen09], which supports two operations (and thereby preserves the ring structure of the plaintexts). The introduction of such new building blocks may spur the development of a new class of voting systems.

In general, we see that developments in voting have followed the following path: 1. identification of notion, 2. refinements, 3. theoretical studies, 4. restart with new notion. Assuming developments continue in this fashion, the question is: where will the focus shift to? What notion will be investigated next?

In the current line of verifiability developments, we observe a move to practical maturity: various of these newly proposed systems have been used in a handful of elections. Such practical exposure raises interesting new questions, concerning user interface design, accessibility, and social aspects (e.g., poll work training, voter education, etc.). Note that these questions are of an ever-increasing interdisciplinary nature, and thus cannot be properly addressed only by computer scientists. In that regard, the community's attention may also turn towards theoretical notions beyond privacy and verifiability, such as fairness (ensuring all voters have equal influence on the result), eligibility (ensuring only those allowed can cast a vote), or accountability (ensuring that errors can be traced back to an offending party).

Finally, the above-noted outreach from the academic community to offer their systems for use in actual elections provides election officials with systems which offer strong guarantees on both privacy and verifiability. It is our hope that this will converge with governmental desire for automation in the election process.

Acknowledgments. This paper would not be in the shape and form it currently is, if not for the extensive comments made by Pascal Lafourcade, for which we are indebted to him. In addition, we are grateful for comments on early drafts of this work by Peter Y. A. Ryan, Marc Pouly, Gabriele Lenzini, and Thea Peacock, as well as for the support of the EPRIV and SerTVS projects and of the Fonds National de la Recherche Luxembourg (FNR).

References

[ACvdG10] R. Araújo, R. F. Custódio, and J. van de Graaf. A verifiable voting protocol based on Farnel. In *Towards Trustworthy Elections*, volume 6000 of *LNCS*, pages 274–288. Springer, 2010.

[Adi08] B. Adida. Helios: Web-based open-audit voting. In *Proc. 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[AdMPQ09] B. Adida, O. de Marneffe, O. Pereira, and Jean-Jacques Quisqater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Proc. EVT/WOTE 2009*. USENIX Association, 2009.

[AFT08] R. Araújo, S. Foulle, and J. Traoré. A practical and secure coercion-resistant scheme for remote elections. In *Frontiers of Electronic Voting*, volume 07311 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany, 2008.

[ALBD04] R. Aditya, B. Lee, C. Boyd, and E. Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In *Proc. 1st Conference on Trust and Privacy in Digital Business*, volume 3184 of *LNCS*, pages 152–161. Springer, 2004.

[AN06] B. Adida and A. Neff. Ballot casting assurance. In *Proc. EVT/WOTE 2006*. USENIX Association, 2006.

[App06] A. W. Appel. How to defeat Rivest's Threeballot voting system. Technical report, Princeton University, 2006.

[AR06] B. Adida and R. L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Proc. ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 29–40. ACM, 2006.

[BCP+11] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. Adapting Helios for provable ballot privacy. In *Proc. 16th European Symposium on Research in Computer Security (ESORICS)*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011.

[Ben94] J. Benaloh. Dense probabilistic encryption. In *Proc. Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[Ben06] J. Benaloh. Simple verifiable elections. In *Proc. Electronic Voting Technology Workshop (EVT) 2006*. USENIX Association, 2006.

[Ben07] J. Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *Proc. Electronic Voting Technology Workshop (EVT) 2007*. USENIX Association, 2007.

[BFP+01] O. Baudron, P. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 274–283. ACM, 2001.

[BG11] K. Bräunlich and R. Grimm. Formalization of receipt-freeness in the context of electronic voting. In *Proc. 6th Conference on Availability, Reliability and Security (ARES'11)*, pages 119–126. IEEE, 2011.

[BHM08] M. Backes, C. Hriţcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st Computer Security Foundations Symposium (CSF)*, pages 195–209. IEEE Computer Society, 2008.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. 1979 National Computer Conference*, volume 48, pages 313–317. American Federation of Information Processing Societies, 1979.

[BMR07] J. Bohli, J. Müller-Quade, and S. Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *Proc. 1st Conference on Voting and Identity (Vote-ID)*, volume 4896 of *LNCS*, pages 111–124. Springer, 2007.

[BRS07] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 62–71. ACM, 2007.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

[CA12] S.-H. Cha and Y. J. An. Taxonomy and nomenclature of preferential voting methods. In *Proc. World Congress on Engineering and Computer Science (WCECS 2012)*, volume 1 of *Lecture Notes in Engineering and Computer Science*, pages 173–178. International Association of Engineers / Newswood Limited, 2012.

[CC97] L. F. Cranor and R. Cytron. Sensus: A security-conscious electronic polling system for the Internet. In *Proc. 30th Annual Hawaii Conference on System Sciences (HICSS)*, pages 561–570. IEEE Computer Society, 1997.

[CCC+08] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Proc. Electronic Voting Technology Workshop (EVT) 2008*. USENIX Association, 2008.

[CCC+10] R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In *Proc. 19th USENIX Security Symposium*, pages 291–306. USENIX Association, 2010.

[CCM08] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *Proc. 29th IEEE Symposium on Security and Privacy (S&P)*, pages 354–368. IEEE Computer Society, 2008.

[CEC+08] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *Proc. 29th IEEE Symposium on Security and Privacy (S&P)*, 6(3):40–46, 2008.

[CFS+06] B. Chevallier-Mames, P. Fouque, J. Stern, D. Pointcheval, and J. Traoré. On some incompatible properties of voting schemes. In *Proc. IAVoSS Workshop On Trustworthy Elections*,

2006.

[CFSY96] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *Proc. 15th Annual Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. 16th Annual Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Proc. 2nd Annual International Cryptology Conference (CRYPTO)*, pages 199–203. Plenum Press, New York, 1982.

[Cha01] D. Chaum. Surevote: Technical overview. In *Proc. Workshop On Trustworthy Elections (WOTE)*, 2001.

[Cha04] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *Proc. 25th IEEE Symposium on Security and Privacy (S&P)*, 2(1):38–47, 2004.

[CRS05] D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A practical voter-verifiable election scheme. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

[CS11] V. Cortier and B. Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *Proc. 24th IEEE Computer Security Foundations Symposium (CSF)*, pages 297–311. IEEE Computer Society, 2011.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DKR06] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proc. 19th Computer Security Foundation Workshop (CSFW)*, pages 28–42. IEEE Computer Society, 2006.

[DKR09] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[DLL11] J. Dreier, P. Lafourcade, and Y. Lakhnech. Vote-independence: A powerful privacy notion for voting protocols. In *4th Canada-France MITACS Workshop on Foundations & Practice of Security (FPS'11)*, volume 6888 of *LNCS*, pages 164–180. Springer, 2011.

[DLL12] J. Dreier, P. Lafourcade, and Y. Lakhnech. A formal taxonomy of privacy in voting protocols. In *First IEEE International Workshop on Security and Forensics in Communication Systems (ICC'12 WS - SFCS)*, pages 6710–6715, 2012.

[DY83] D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[ECA08] A. Essex, J. Clark, and C. Adams. Aperio: High integrity elections for developing countries. In *Proc. IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2008.

[ECCP07] A. Essex, J. Clark, R. Carback, and S. Popoveniuc. Punchscan in practice: An E2E election case study. In *Proc. IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2007.

[ECHA10] A. Essex, J. Clark, U. Hengartner, and C. Adams. Eperio: mitigating technical complexity in cryptographic election verification. In *Proc. EVT/WOTE 2010*. USENIX Association, 2010.

[ED10] S. Estehghari and Y. Desmedt. Exploiting the client vulnerabilities in internet e-voting systems: Hacking Helios 2.0 as an example. In *Proc. EVT/WOTE 2010*. USENIX Association, 2010.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. 5th Annual International Cryptology Conference (CRYPTO)*, volume 218 of *LNCS*, pages 10–18. Springer, 1985.

[FCS06] K. Fisher, R. Carback, and A. T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *Proc. IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2006.

[FG07] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:1–10, 2007.

[FLA11] L. Fousse, P. Lafourcade, and M. Alnuaimi. Benaloh's dense probabilistic encryption revisited. In *Proc. 4th International Conference on Progress in Cryptology in Africa (AFRICACRYPT'11)*, volume 6737 of *LNCS*, pages 348–362. Springer, 2011.

[FMM+02] J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Proc. 12th Conference on Financial Cryptography and Data Security (FC)*, volume 2357 of *LNCS*, pages 16–30. Springer, 2002.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proc. Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. 6th Annual International Cryptology Conference (CRYPTO)*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[Gen09] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[Her97] M. A. Herschberg. Secure electronic voting over the World Wide Web. Master's thesis, Massachusetts Institute of Technology, 1997.

[Hir01] M. Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, 2001.

[HRT10] J. Heather, P. Y. A. Ryan, and V. Teague. Pretty good democracy for more expressive voting schemes. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS)*, volume 6345 of *LNCS*, pages 405–423. Springer, 2010.

[HS00] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. 19th Annual Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[HS07] J. Helbach and J. Schwenk. Secure internet voting with code sheets. In *Proc. 1st Conference on E-Voting and Identity (Vote-ID)*, volume 4896 of *LNCS*, pages 166–177. Springer, 2007.

[JCJ05] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proc. ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 61–70. ACM, 2005.

[JdV06] H. L. Jonker and E. P. de Vink. Formalising receipt-freeness. In *Proc. 9th Conference on Information Security (ISC)*, volume 4176 of *LNCS*, pages 476–488. Springer, 2006.

[JJR02] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. 11th USENIX Security Symposium*, pages 339–353. USENIX Association, 2002.

[JMP09a] H. L. Jonker, S. Mauw, and J. Pang. A formal framework for quantifying voter-controlled privacy. *Journal of Algorithms in Cognition, Informatics and Logic*, 64(2-3):89–105, 2009.

[JMP09b] H. L. Jonker, S. Mauw, and J. Pang. Measuring voter-controlled privacy. In *Proc. 4th Conference on Availability, Reliability and Security (ARES)*, pages 289–298. IEEE Computer Society, 2009.

[JP06] H. L. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proc. IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2006.

[JP11] H. L. Jonker and J. Pang. Bulletin boards in voting systems: Modelling and measuring privacy. In *Proc. 6th Conference on Availability, Reliability and Security (ARES)*, pages 294–300. IEEE Computer Society, 2011.

[JR12] R. Joaquim and C. Ribeiro. An efficient and highly sound voter verification technique and its implementation. In *Proc. 3rd conference on E-Voting and Identity (VoteID'11)*, volume 7187 of *LNCS*, pages 104–121. Springer, 2012.

[JRF09] R. Joaquim, C. Ribeiro, and P. Ferreira. VeryVote: A voter verifiable code voting system. In *Proc. 2nd Conference on Voting and Identity (Vote-ID)*, volume 5767 of *LNCS*, pages 106–121. Springer, 2009.

[JRF10] R. Joaquim, C. Ribeiro, and P. Ferreira. Improving remote voting security with codevoting. In *Towards Trustworthy Elections*, volume 6000 of *LNCS*, pages 310–329. Springer, 2010.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proc. 15th Annual Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, pages 143–154, 1996.

[KR05] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Proc. 14th European Symposium on Programming (ESOP)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[KRMC10] J. Kelsey, A. Regenscheid, T. Moran, and D. Chaum. Attacking paper-based e2e voting

systems. In *Towards Trustworthy Elections*, volume 6000 of *LNCS*, pages 370–387. Springer, 2010.

[KRS10] S. Kremer, M. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS)*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[KT09] R. Küsters and T. Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. In *Proc. 30th IEEE Symposium on Security and Privacy (S&P)*, pages 251–266. IEEE Computer Society, 2009.

[KTV10] R. Küsters, T. Truderung, and A. Vogt. A game-based definition of coercion-resistance and its applications. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF)*, pages 122–136. IEEE Computer Society, 2010.

[LBD+04] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. 5th Conference on Information and Communications Security (ICICS)*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.

[LK00] B. Lee and K. Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *Proc. Joint workshop on Information Security and Cryptology*, pages 101–108. Institute of Electronics, Information and Communication Engineers in Japan, 2000.

[LK02] B. Lee and K. Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Proc. 5th Conference Information Security and Cryptology (ICISC)*, volume 2587 of *LNCS*, pages 389–406. Springer, 2002.

[MB01] S. Mauw and V. Bos. Drawing Message Sequence Charts with LaTeX. *TUGBoat*, 22(1-2):87–92, March/June 2001.

[MBC01] E. Magkos, M. Burmester, and V. Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *Proc. IFIP Conference on Towards The E-Society: E-Commerce, E-Business, E-Government*, volume 202 of *IFIP Conference Proceedings*, pages 683–694. Kluwer, 2001.

[MH96] M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *Proc. 4th Conference on the Theory and Applications of Cryptology (ASIACRYPT)*, volume 1163 of *LNCS*, pages 125–132. Springer, 1996.

[MN06] T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Proc. 26th Annual International Cryptology Conference (CRYPTO)*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.

[MN07] T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. In *Proc. 14th ACM Conference on Computer and Communications Security (CCS)*, pages 246–255. ACM, 2007.

[MVdV07] S. Mauw, J.H.S. Verschuren, and E.P. de Vink. Data anonymity in the FOO voting scheme. In *Proc. 2nd International Workshop on Views On Designing Complex Architectures (VODCA)*, volume 168 of *ENTCS*, pages 5–28, 2007.

[Nef01] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS)*, pages 116–125. ACM, 2001.

[Nef04] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Votehere, Inc. whitepaper, 2004. `http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.134.1006&rep=rep1&type=pdf` (last accessed 2 April 2012).

[NR94] V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In *Proc. 2nd Conference on the Theory and Applications of Cryptology (ASIACRYPT)*, volume 917 of *LNCS*, pages 164–170. Springer, 1994.

[Oka96] T. Okamoto. An electronic voting scheme. In *Proc. IFIP World Conference on IT Tools*, pages 21–30. Chapman & Hall, 1996.

[Oka97] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 25–35. Springer, 1997.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. 18th Annual Conference on the Theory and Applications of Cryptographic Techniques: Cryptology (EUROCRYPT)*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[Ped91] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. 11th Annual International Cryptology Conference (CRYPTO)*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[PV08] S. Popoveniuc and P. Vora. A framework for secure electronic voting. In *Proc. IAVoSS Workshop On Trustworthy Elections (WOTE)*, 2008.

[RS07] R. L. Rivest and W. D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. In *Proc. USENIX Electronic Voting Technology Workshop*. USENIX Association, 2007.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Rya05] P. Y. A. Ryan. A variant of the Chaum voter-verifiable scheme. In *Proc. Workshop on Issues in the Theory of Security (WITS)*, pages 81–88. ACM, 2005.

[SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proc. 18th IEEE Symposium on Security and Privacy (S&P)*, pages 44–54. IEEE Computer Society, 1997.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In *Proc. 14th Annual Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

[SKHS11] O. Spycher, R. Koenig, R. Haenni, and M. Schläpfer. A new approach towards coercion-resistant remote e-voting in linear time. In *Proc. 15th Conference on Financial Cryptography and Data Security (FC)*, volume 7035 of *LNCS*, pages 182–189. Springer, 2011.

[SLC+11] S. Schneider, M. Llewellyn, C. Culnane, J. Heather, S. Srinivasan, and Z. Xia. Focus group views on Prêt à voter 1.0. In *Proc. Workshop on Requirements Engineering for Electronic Voting Systems (REVOTE)*, pages 56–65. IEEE Computer Society, 2011.

[Smi05a] W. D. Smith. New cryptographic voting scheme with best-known theoretical properties. In *Proc. Workshop on Frontiers in Electronic Elections (FEE)*, 2005.

[Smi05b] Warren D Smith. Cryptography meets voting, September 2005.

[SP05] K. Sampigethaya and R. Poovendran. A framework and taxonomy for comparison of electronic voting schemes. *Computers & Security*, 25(2):137–153, 2005.

[TMT+08] M. Talbi, B. Morin, V. Viet Triem Tong, A. Bouhoula, and M. Mejri. Specification of electronic voting protocol properties using ADM logic: FOO case study. In *Proc. 10th Conference on Information and Communications Security (ICICS)*, volume 5308 of *LNCS*, pages 403–418. Springer, 2008.

[Uni05] United States Election Assistance Commission. Voluntary Voting Systems Guidelines, 2005.

[vdG09] J. van de Graaf. Voting with unconditional privacy by merging Prêt à Voter and PunchScan. *IEEE Transactions on Information Forensics and Security*, 4(4):674–684, 2009.

[WAB07] S. G. Weber, R. Araújo, and J. Buchmann. On coercion-resistant electronic elections with linear work. In *Proc. 2nd Conference on Availability, Reliability and Security (ARES)*, pages 908–916. IEEE Computer Society, 2007.

[Wik03] D. Wikström. Five practical attacks for "optimistic mixing for exit-polls". In *Proc. 10th Workshop on Selected Areas in Cryptography (SAC)*, volume 3006 of *LNCS*, pages 160–175. Springer, 2003.