End-to-End Verifiable Elections with Casting of Publicly Audited, Cleartext Ballots

Peter Y A Ryan[1], Peter B Roenne[1], Afonso Arriaga[1], and Olivier Pereira[2]

Abstract:

Many end-to-end verifiable schemes have been proposed, and these typically require voters to perform some form of ballot audit in order to achieve cast-as-intended assurance. Such audits usually take the form of a cut-and-choose procedure, often in sequential form, where voters randomly decide at each stage whether to audit or cast (*Benaloh challenges*). These are generally thought to be awkward from a usability point of view and hard to motivate and understand.

Here, we propose an in-person scheme in which ballot audits are outsourced to independent auditors and observers. There is no longer any need or expectation that voters perform ballot audits (but the option to audit can still be offered to them). All ballots are audited, including ones that are used to cast votes, thus there is no cut-and choose needed, resulting in high levels of ballot assurance.

This contributes to improved usability and acceptability. It also means that the assurance in the outcome is no longer reliant on a good proportion of voters performing ballot audits with sufficient diligence, the outcome of the election is verified even if no voter performs a ballot audit. Furthermore, ballot audits take the form of verifying cryptographic signatures rather than verifying encryptions, avoiding the need to reveal randomization factors, etc. For transparency and usability, the ballots display a clear text version of the vote to the voter. Dispute resolution is improved as ballot auditing is simply signature verification.

1 Introduction

Designing a secure election system is arguably one of the greatest challenges facing the information security community: requiring reconciling the contending requirements of transparency and (vote) privacy in the face of adversaries ranging from a coercive spouse to a hostile nation state. Ideally, this should be achieved with minimal trust assumptions. Furthermore, it must be supremely accessible, usable, and understandable by the entire electorate. A good voting system should not only provide the right outcome (winners), but also provide sufficient evidence to prove the correctness of the outcome to all stakeholders.

This motivates the investigation of *End-to-End Verifiable* (E2E-V) schemes that aim to make elections more trustworthy by allowing voters and observers to perform checks to confirm that votes are correctly received, recorded, and tallied. Such schemes aim to ensure that any manipulation of ballots, from the moment they are cast to the point at which they are tallied, is detectable. At a high level of abstraction, most schemes work as follows:

[1] University of Luxembourg,
[2] UCLouvain,

1. An encryption or encoding of the vote is created, usually at the time of casting, which is posted on a bulletin board BB. The voter retains a copy allowing them to check that their ballot appears correctly on the BB.
2. Once voting is over, the set of posted encrypted ballots on the BB is processed to extract the outcome in a verifiable manner while preserving vote privacy.

Many schemes have been proposed following this paradigm, and some developed and trialed, but to date there has been no continued deployment in government elections. This prompts the question: why is there such low acceptance of such systems? It seems probable that a major factor is the need for voters to perform various checks and audits whose motivation may be unclear. Here we explore ways to achieve E2E-V, indeed more: ensuring that elections are not just verifiable, but in fact **verified**, without needing voters to perform checks.

The underlying philosophy behind the E2E-V approach is to make voters the foundation of trust: enabling them to monitor the correct execution of the election. This is a laudable goal, but has proved problematic and probably a major obstacle to deployment. In practice, few voters are motivated to perform the various checks and prefer to just vote and go. In schemes in which some checks are "individual", i.e. can be performed only by the voter, this will typically result in elections that cannot be deemed to have been verified, even if the underlying scheme is verifiable.

Here, we shift the foundation of trust from the voters to independent auditors. This is closer to conventional voting systems with observers, but with the crucial difference: in our scheme what is observed, audited, etc. are not ephemeral processes but rather enduring evidence available to all observers.

We should point out that our approach gives a form of E2E V that is conditional on certain assumptions, in contrast to many schemes that give an unconditional form of E2E V. We detail the assumptions later, but they include: signing keys are not compromised and the signatures committed in the ballot packs are not leaked during the voting period. This is a downside, but we believe that the assumptions are reasonable and the benefits outweigh the need for the additional trust. Note that we still allow voters to perform ballot audits in the usual way so we still have the underpinning of unconditional E2E V.

While conventional E2E V can give unconditional guarantees to the voter, the guarantee in individual: that their own vote makes it into the tally correctly. In itself, this is not enough: people seek assurance that the tally is correct. We have heard this concern from colleagues and in focus groups, etc., and we suspect that it is one of the obstacles to uptake of E2E V. The usual response is: all other voters can perform the checks so it is verifiable. But psychologically and rationally this may not be really compelling, especially when one realises that in reality a small proportion of voters do the checks. Our approach makes these checks, and hence the resulting guarantees, essentially universal.

A key element of E2E-V is to provide voters with confidence that their vote is correctly encrypted, often referred to as *cast-as-intended assurance* or *ballot assurance*. This aspect is the focus of this paper. Typically, such assurance is provided by requiring, at the time of voting, the voting device to commit to an encryption of the vote which can then be audited. Auditing a ballot typically involves revealing randomisation factors to an independent device which jeopardises receipt-freeness. Consequently, it has long been assumed that some form of cut-and-choose mechanism is required: audited ballots are discarded and non-audited ballots cast.[3] Typically, this is done in a sequential manner, e.g. *Benaloh challenges*: the device commits to an encryption and the voter decides either to audit or to cast this [Be06]. If the voter opts to audit, a fresh encryption is produced and the voter again has the choice to audit or cast, and so on until the voter is content and casts the last, non-audited encryption.

A notable exception to the use of a cut-and-choose is Neff's MarkPledge scheme, where the audited ballot is cast [Ne04]. This is an in-person scheme that involves an interactive ZK proof of the encrypted vote between the voting device and the voter. Receipt-freeness is restored by masking the real, interactive proof performed in the booth with fake transcripts of proofs for the other candidates. Unfortunately, this ingenious and elegant scheme did not take off commercially, presumably due to the usability issues.

Cut-and-choose ballot audits place a burden on voters for which voters (and stakeholders) struggle to understand the motivation [Ka11]. The extent to which an election can be deemed to have been verified, as opposed to just verifiable, depends on the extent to which voters are diligent in performing the audits and reporting anomalies, etc. A further downside of such cut-and-choose mechanisms is that the level of (ballot) assurance they provide is rather modest: a corrupt device has a nonnegligible chance of cheating undetected. Benaloh challenges also present potential privacy issues: for an audited ballot, the vote is revealed. Of course, a voter who intends to audit and spoil the ballot should input a random vote at this stage and only their intended vote when they plan to cast, but it is not clear that many voters appreciate this. Furthermore, if a corrupt encrypting device has some indication of the voter's preference, it may exploit this to improve the odds of its cheating strategy. Dispute resolution may also be problematic in Benaloh challenges in that, in the event of a challenge, it may be hard to distinguish between a cheating device and a voter lying about or misremembering what vote was input.

Cut-and-choose procedures are necessary if the ciphertext that appears on the BB is the same ciphertext that is audited. However, if we re-encrypt the ciphertext before posting, then the secrecy of the encryption is restored. We still want the voter to be able to identify their ballot on the BB and be confident that the re-encryption is performed correctly, i.e. leaving the vote/plaintext invariant. Malleable signatures are one primitive that can achieve this: a malleable signature is applied to the ballot which is transformed but remains valid if and only if the posted ciphertext is indeed a re-encryption of the submitted ballot as in e.g. [Ch16]. Another possibility is to have the voter participate in an interactive ZK proof of

[3] The Estonian voting system supports auditing cast ballots, but these are not posted to a public bullet board [CM17].

re-encryption with the entity that performs the re-encryption. The latter complicates the voting experience and impacts usability in a similar manner to MarkPledge. Finally, a more recent solution is traceable encryption, TREnc [DPP22], which we will discuss below.

The other kind of check that voters are typically expected to perform in E2E-V systems is the presence of their ballot on the BB. However, if a well-curated, independent record or records of cast ballots is kept, it is possible to outsource such checks as well. For example, at the time of casting, the ballot slips could be fed into a scanner that automatically feeds the slips into a ballot box, for later verification of consistency with the BB. Independent entities could also keep a log of the signing keys used to cast ballots.

Thus, it appears possible to have E2E-verified elections where the checks that are usually thought of as necessarily performed by the voter, that is, "individual", become "universal". This does come at the cost of somewhat stronger trust assumptions, but in many contexts this may be justified and should increase the acceptability of evidence-based elections.

Tracker-based schemes such as Selene [RRI16] and Hyperion [Da24] remove the need for voters to identify their (encrypted) receipt on the BB as they can check their vote in plaintext in the tally. This suggests the possibility that for such schemes we could simply post the re-encrypted ballots without needing malleable signatures or similar primitives, before these ballots are shuffled and decrypted for plaintext verification. This was also used in [Mo24] to achieve everlasting receipt-freeness.

1.1 We Audit the Ballots So You Don't Have To

For schemes involving on-the-fly encryption of the vote, only the voter should know the intent, and so only they can check that the intent is respected. However, for schemes that involve pre-computation and commitment of encryptions we can use re-encryption techniques to allow pre-auditing of ballots, enabling the outsourcing of ballot audits. The possibility of pre-committing and pre-auditing encrypted ballots first appears in Prêt-à-Voter, [Ry09], and has resurfaced in other schemes since.

A further observation, that prompts a key innovation of our approach, is that, if ballots are re-encrypted before posting, the pre-committed ballots can be trivial encryptions of the options, i.e. with randomisation zero. This means that the vote options can be shown to the voter in the clear and is thus immediately and humanly verifiable, while the device might use OCR to interpret it and a QR code to capture the associated cryptographic material, such as a malleable signature. This is an improvement in transparency compared to [CFL19] where voters have to check an encryption using its randomness, but with the downside that the booth device learns the vote. We discuss this trade-off in Section 2.

Our goal then is to ensure that all ballots are audited by independent entities, while keeping the option for voters to additionally perform their own audits. We provide a detailed description of a particular scheme in Section 4, but to sketch the key ideas: In the setup

phase, ballot packs are generated; Each pack has its own unique signing key pair (sk_j, vk_j) and will comprise (trivial) encryptions of the various options, each on a separate slip of paper and malleably signed with sk_j. We assume that it is architecturally and procedurally enforced that the scanner in the voting booth never sees any of these signing keys.

In the booth, the voter presents the slip showing their choice to a scanner.[4] The booth device re-encrypts the encrypted term and transforms the signature accordingly and prints the result. Election officials check that the transformed signature is valid and, if so, the signature and encryption are scanned and posted to the BB. The voter can identify her ballot on BB using the signature.

2 Contribution and Related Work

We show that, by combining several observations, some known and some novel, E2E-V can be achieved with all ballots being audited without requiring voters to perform ballot audits. Cast-as-intended is not only verifiable, but in fact verified.

The above goals can be achieved in various ways, using various primitives, but for concreteness we propose a specific, in-person scheme. The scheme we describe here uses malleable signatures, but other primitives that allow the voter to track their transformed ballot on BB, such as TREncs described in the next section, could also be used.

Here we focus on an in-person scheme based on these observations. Achieving such goals for online/remote voting presents very different challenges that will be addressed elsewhere. In person has the advantage that we can exploit the special properties of paper ballots allowing us to invoke milder assumptions than necessary in a remote context where we have, for example, to worry about devices displaying information to the voter that is different from what is processed internally. Paper has the further advantage of potentially generating a paper audit trail and the use of RLAs etc. for greater assurance.

The key features of our approach are:

(*i*) Ballot audits are performed by independent entities rather than by the voters themselves.

(*ii*) Our ballot forms have trivial encryptions that allow the vote to be displayed to the voter in plaintext.

(*iii*) Ballot audits are simple signature verifications and can be public.

(*iv*) All ballots can be audited, both cast and uncast (avoiding the need for cut-and-choose protocols).

[4] Potentially the voters could apply their own malleable signature to the encryption, but this requires more infrastructure, complicates the voting ceremony and may introduce threats.

In effect, we turn ballot auditing, usually thought of as necessarily individual, to being universal.

Note that although in our approach there is now no requirement for voters to perform ballot audits, they can still be offered the opportunity to do this if they desire. The point is that a good level of assurance of cast-as-intended is not dependent on a sufficient number of voters performing ballot audits in a diligent fashion, reporting issues, etc. Ideally, voters should also check the validity of the signature over the candidate, but checking a signature, requiring only knowledge of the public verification key, is much simpler than checking an encryption.

The closest work to ours is the BeleniosVS scheme of Cortier et al. [CFL19], which also uses paper ballots and re-encrypts ballots before posting, but the purpose is to achieve receipt-freeness rather than remove the need for voters to perform ballot audits. Their ballots use regular encryptions (i.e., with uniform randomness), which has the advantage that the entity that performs the re-encryption does not learn the vote, but is less transparent to the voter and audits require opening the encryption. This results in a trade-off between BeleniosVS and our scheme. We argue that as long as the assumption of minimal functionality and connectivity of the booth device can be enforced, then its fleeting ability to learn the vote should be deemed acceptable, at least in some contexts.

We also note that BeleniosVS requires that voters possess passwords to authenticate to the server. In our scheme, no passwords are required; voters simply authenticate at the polling station in the conventional way.

Finally, the goals of our approach can also be achieved using more general methods than malleable signatures, e.g., using traceable encryption, as we will discuss.

3 Crypto Primitives

Randomisable encryption allows anyone to transform an existing ciphertext (without knowledge of the underlying plaintext) into a new ciphertext that encrypts the same plaintext and has the same distribution as a fresh encryption of that plaintext. This has multiple applications, but has been particularly useful in the context of e-voting for the ability to anonymise an encrypted ballot. ElGamal encryption is the most notable example of an encryption scheme exhibiting such a randomisation property.

Our voting scheme requires a signature mechanism that can be used to produce a signature of a plaintext, or of a trivial encryption of a plaintext, in such a way that a voting device can later transform that signature into a valid signature of a regular encryption of the same plaintext and of the same plaintext only. Ciphertexts on BB can be identified by the verification key with which they are signed (and fingerprinting of the verification key). Vote privacy is preserved as all slips within a given ballot package are signed with the same signature key.

Various mechanisms have been designed to offer this functionality. Devillez et al. formalized the associated requirement and proposed a new primitive called Traceable Receipt-free Encryption (TREnc), which offers a way to build receipt-free voting schemes in a generic way [DPP22]. Adversarily chosen TREnc ciphertexts can be rerandomised by a third party in such a way that the adversary becomes unable to demonstrate what plaintext a rerandomised ciphertext actually encrypts, while making sure that the third-party is unable to modify that plaintext. In order to guarantee ballot privacy, TREnc ciphertexts also offer a form of non-malleability that is sufficient to prevent ballot copies (something that Signatures on Randomisable ciphertexts do not guarantee, for instance). As such, a TREnc mechanism would suit our needs. However, even trivial TREnc ciphertexts may remain relatively large to encode in a single QR code, and we observe that TREnc actually offers more than we need: in our case, there is no privacy requirement with respect to the booth device that performs the re-encryption, and could potentially take advantage of the knowledge of the plain-text to obtain more efficient solutions.

We now turn to a more lightweight primitive known as Signatures on Randomisable Ciphertexts (SoRC) [Bl11] used as an ingredient in multiple other voting schemes [CFL19; CGG19; Ch16; HPP20]: a SoRC brings the possibility of signing ciphertexts that can be rerandomised while keeping the signatures valid. A SoRC does not bring all the properties that we need though: in particular, they lack the form of non-malleability that would prevent an adversarial authority from "copying" a cast ciphertext by rerandomising the ciphertext and signing it with another key, which would in turn break ballot privacy. But because the booth device knows the randomness of the ciphertext, we can adopt standard techniques and simply add a proof of Σ knowledge of that randomness. If the SoRC offers ciphertexts such that the randomisation of a trivial ciphertext is IND-CPA secure, then the addition of the Σ-proof provides a non-malleable (NM-)CPA ciphertext, which is sufficient for ballot privacy [BPW12]. The form of IND-CPA security coming from rerandomised ciphertexts is actually provided if we use a Fully Class-Hiding encryption mechanism, as defined by Bauer et al. [BF20]. Finally, we suspect that more efficient solutions could be found by taking advantage of the fact that the only ciphertexts that need to be (re-)randomised are trivial ones, but we leave that question open for now. We now recall the syntax, correctness, and security properties of SoRC [BF20].

Definition 1 *A SoRC scheme combines a randomisable public-key encryption scheme with a signature scheme. A Signatures on Randomisable Ciphertext scheme* SoRC *is defined by a tuple of polynomial-time algorithms* $\mathsf{SoRC} = (\mathsf{Setup}, \mathsf{EKeygen}, \mathsf{SKeygen}, \mathsf{Enc}, \mathsf{Rerand}, \mathsf{Dec}, \mathsf{Sign}_M, \mathsf{Verify}, \mathsf{Adapt})$ *that behave as follows:*

- $\mathsf{Setup}(\lambda) \rightarrow \mathsf{pp}$*: a probabilistic setup algorithm that on input a security parameter* λ*, outputs the public parameters* pp*, including elliptic curves, groups, group generators and bilinear maps.*
- $\mathsf{EKeygen}(\mathsf{pp}) \rightarrow (\mathsf{dk}, \mathsf{ek})$*: a probabilistic key generation algorithm for encryption*

that when entering the public parameters pp *outputs a pair of keys* $(\mathsf{dk}, \mathsf{ek})$ *for decryption and encryption.*

- $\mathsf{SKeygen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$*: a probabilistic key generation algorithm for signature that when entering the public parameters* pp *outputs a pair of keys* $(\mathsf{sk}, \mathsf{vk})$ *for signing and verification.*
- $\mathsf{Enc}(\mathsf{ek}, \mathsf{m}; \mathsf{r}) \to \mathsf{c}$*: an encryption algorithm that, given a public encryption key* ek*, a plaintext* m *and possibly some explicit random coins* r *outputs a ciphertext* c*. The algorithm is deterministic as* r *is passed as an explicit argument.*
- $\mathsf{Rerand}(\mathsf{ek}, \mathsf{c}, \mathsf{r}) \to \mathsf{c}'$*: a deterministic randomization algorithm that on input of a public encryption key* ek*, a ciphertext* c *and random coins* r *outputs a new ciphertext with the same plaintext* c'*.*
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c}) \to \mathsf{m}$*: a deterministic decryption algorithm that on input a secret decryption key* dk *and a ciphertext* c *outputs a plaintext message* m*.*
- $\mathsf{Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c}) \to \sigma$*: a (possibly probabilistic) signing algorithm that on input a signing key* sk*, an encryption key* ek *and a ciphertext* c *outputs a signature* σ*.*
- $\mathsf{Adapt}(\sigma, \mathsf{r}) \to \sigma'$*: a (possibly probabilistic) signature-adaptation algorithm that on input a signature* σ *and* r *outputs an adapted signature* σ'*. We stress that signature-adaptation (see below) may be randomised and that* r *is simply an input.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma) \to \{0, 1\}$*: a deterministic signature-verification algorithm that on input a verification key* vk*, an encryption key* ek*, a ciphertext* c *and a signature* σ *outputs either* 0 *or* 1*.*

SoRC correctness requires the encryption and signature schemes to be correct and the adapted signatures to verify successfully.

Definition 2 *A SoRC scheme is correct if for any public parameter* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$*, for all pairs* $(\mathsf{dk}, \mathsf{ek}) \leftarrow\!\!\$\ \mathsf{EKeygen}(\mathsf{pp})$ *and* $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{SKeygen}(\mathsf{pp})$*, for all messages* $\mathsf{m} \in \mathbb{M}$*,* $\mathsf{r} \in \mathbb{R}$ *and* $\mathsf{c} \in \mathbb{C}$*, we have that:*

1. $\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, \mathsf{m}; \mathsf{r})) = \mathsf{m}$
2. $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \mathsf{Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c})) = 1$
3. $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma) = 1 \implies \mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}, \mathsf{r}), \mathsf{Adapt}(\sigma, \mathsf{r})) = 1$

Security properties. We are interested in four properties of signatures on randomisable ciphertexts: two related to the underlying randomisable public-key encryption scheme and two related to the malleable signature.

For the underlying randomisable public-key encryption, the two properties of interest are: (i) the standard security notion known as indistinguishability under chosen-plaintext attacks;

and (ii) fully class-hiding, a notion originally from signature on equivalence classes [FHS14; HS14], but adapted to randomisable encryption and strengthened as in [BF20] so that the adversary chooses (selects or maliciously crafts) the ciphertext before randomization. It should be noted that, while a ciphertext with trivial encryption is not maliciously crafted, this scenario falls outside the scope of the original definition in [Bl11]. Allowing the adversary to select the ciphertext aligns well with our e-voting scheme, as the adversary could deliberately construct the ciphertext using the trivial randomisation we employ.

We omit the definition of IND-CPA security–which is standard–and we remind the fully class hiding property of an encryption scheme.

Definition 3 ***(class-hiding)*** *A SoRC scheme is said to be* CL-HID *secure if for any PPT adversary $\mathcal{A}$ engaged in the* CL-HID *security game, where $\mathcal{A}$ is a two stage adversary, the advantage of $\mathcal{A}$ defined as:*

$$\mathsf{Adv}^{\text{CL-HID}}_{\text{SoRC},\mathcal{A}}(\lambda) \;:=\; 2 \cdot \Pr[\text{CL-HID}^{\mathcal{A}}_{\text{SoRC}}(\lambda) = 1] - 1 \tag{1}$$

is a negligible function of the security parameter λ. Experiment CL-HID *is defined in Fig. 1.*

$\text{Exp CL-HID}^{\mathcal{A}}_{\text{SoRC}}(\lambda)$	*(continued)*
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$	$\mathsf{c}_0 \leftarrow \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}, \mathsf{r})$
$(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EKeygen}(\mathsf{pp})$	$\mathsf{c}_1 \leftarrow\!\!\$\ \mathbb{C}$
$b \leftarrow\!\!\$\ \{0,1\}$	$b' \leftarrow \mathcal{A}_1(st, \mathsf{c}_b)$
$(\mathsf{c}, st) \leftarrow \mathcal{A}_0(\mathsf{ek})$	**return** $b == b'$
$\mathsf{r} \leftarrow\!\!\$\ \mathbb{R}$	

Fig. 1: Fully Class-Hiding (CL-HID). A legitimate adversary $\mathcal{A}$ against CL-HID outputs $\mathsf{c} \in \mathbb{C}$.

We are also interested in two properties of malleable signatures: (i) the standard notion of existential unforgeability adapted to malleable signatures (UF-CMA security) and (ii) the distribution of signature adaptation under malicious keys.

Definition 4 ***(existential unforgeability under chosen-message attack)*** *A SoRC scheme is said to be* UF-CMA *secure if for any PPT adversary $\mathcal{A}$ engaged in the* UF-CMA *security game, the advantage of $\mathcal{A}$ defined as:*

$$\mathsf{Adv}^{\text{UF-CMA}}_{\text{SoRC},\mathcal{A}}(\lambda) \;:=\; \Pr[\text{UF-CMA}^{\mathcal{A}}_{\text{SoRC}}(\lambda) = 1] \tag{2}$$

is a negligible function of the security parameter λ. Experiment UF-CMA *is defined in Fig. 2.*

Definition 5 ***(signature-adaptation under malicious keys)*** *A SoRC scheme is said to be* ADAPT *secure if for any unbounded adversary $\mathcal{A}$ engaged in the* ADAPT *security game, where $\mathcal{A}$ is a two stage adversary, the advantage of $\mathcal{A}$ defined as:*

$$\mathsf{Adv}^{\mathsf{ADAPT}}_{\mathsf{SoRC},\mathcal{A}}(\lambda) \;:=\; 2 \cdot \Pr[\mathsf{ADAPT}^{\mathcal{A}}_{\mathsf{SoRC}}(\lambda) = 1] - 1 \tag{3}$$

is a negligible function of the security parameter λ. Experiment ADAPT *is defined in Fig. 2.*

$\mathsf{Exp\ UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SoRC}}(\lambda)$	$\mathsf{Exp\ ADAPT}^{\mathcal{A}}_{\mathsf{SoRC}}(\lambda)$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$	$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKeygen}(\mathsf{pp})$	$(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma, \mathsf{r}, st) \leftarrow \mathcal{A}_0(\mathsf{pp})$
$(\mathsf{ek}, \mathsf{c}, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_M(\mathsf{sk},\cdot,\cdot)}(\mathsf{vk})$	$\sigma_0 \leftarrow\!\!\$\ \mathsf{Adapt}(\sigma, \mathsf{r})$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma) == 1 \wedge (\mathsf{ek}, \mathsf{c}) \notin \mathsf{List}$	$\mathsf{c}^* \leftarrow \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}, \mathsf{r})$
	$\sigma_1 \leftarrow\!\!\$\ \{\sigma^* \mid \mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}^*, \sigma^*)\}$
$\mathsf{Oracle\ Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c})$	$b \leftarrow\!\!\$\ \{0, 1\}$
$\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{ek}, \mathsf{c}^*) \mid \mathsf{c}^* \sim \mathsf{c} \text{ under } \mathsf{ek}\}$	$b' \leftarrow \mathcal{A}_1(st, \sigma_b)$
return $\mathsf{Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c})$	**return** $b == b'$

Fig. 2: Security experiments defining properties of SoRC related to the underlying malleable signature: (i) Existential unforgeability under chosen message attack (UF-CMA) for SoRC; (ii) signature-adaptation under malicious keys (ADAPT). When the adversary $\mathcal{A}$ against UF-CMA queries the Sign_M oracle with a pair $(\mathsf{ek}, \mathsf{c})$, all pairs $(\mathsf{ek}, \mathsf{c}^*)$ such that c^* encrypts the same message as c under ek are added to List. The adversary wins if it forges a valid signature for a ciphertext not in List. A legitimate adversary $\mathcal{A}$ against ADAPT outputs $(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma)$ such that $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}, \sigma) == 1$. ADAPT is a statistical notion, therefore $\mathcal{A}$ against ADAPT is unbounded.

Class hiding and signature adaptation ensure that the re-encrypted and adapted ciphertext-signature pair of a ciphertext with trivial encryption of $\mathrm{Candidate}_0$ is indistinguishable from the re-encrypted and adapted ciphertext-signature pair of a ciphertext with a trivial encryption of $\mathrm{Candidate}_1$.

$$\begin{aligned}
&\mathsf{c}_0 = \mathsf{Enc}(\mathsf{ek}, \mathrm{Candidate}_0; 0), \quad \sigma_0 = \mathsf{Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c}_0)\\
&\mathsf{c}_1 = \mathsf{Enc}(\mathsf{ek}, \mathrm{Candidate}_1; 0), \quad \sigma_1 = \mathsf{Sign}_M(\mathsf{sk}, \mathsf{ek}, \mathsf{c}_1)\\
&\mathsf{r} \leftarrow\!\!\$\ \mathbb{R}
\end{aligned}$$

$$\begin{aligned}
& (\mathsf{c}^* = \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}_0, \mathsf{r}), \quad \sigma^* = \mathsf{Adapt}(\sigma_0, \mathsf{r}))\\
\sim_{\mathsf{ADAPT}}\ & (\mathsf{c}^* = \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}_0, \mathsf{r}), \quad \sigma^* \leftarrow\!\!\$\ \{\sigma \mid \mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}^*, \sigma)\})\\
\sim_{\mathsf{CL\text{-}HID}}\ & (\mathsf{c}^* \leftarrow\!\!\$\ \mathbb{C}, \quad \sigma^* \leftarrow \{\sigma \mid \mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}^*, \sigma)\})\\
\sim_{\mathsf{CL\text{-}HID}}\ & (\mathsf{c}^* = \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}_1, \mathsf{r}), \quad \sigma^* \leftarrow\!\!\$\ \{\sigma \mid \mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \mathsf{c}^*, \sigma)\})\\
\sim_{\mathsf{ADAPT}}\ & (\mathsf{c}^* = \mathsf{Rerand}(\mathsf{ek}, \mathsf{c}_1, \mathsf{r}), \quad \sigma^* = \mathsf{Adapt}(\sigma_1, \mathsf{r}))
\end{aligned}$$

Replacing a ballot in the BB with one for a chosen candidate, without knowledge of the pre-committed signing key, would require forging a signature.

Practical instantiation. We can instantiate our e-voting scheme with the highly efficient bilinear pairing SoRC scheme from [BF20], which has been proven to satisfy all the security properties required in the generic group model. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ represent the bilinear map in the SoRC scheme from [BF20]. A ciphertext/signature pair consists of 6 group elements: the ElGamal ciphertext includes 2 elements in $\mathbb{G}_1$, while the malleable signature includes 3 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. Using the Barreto-Lynn-Scott pairing-friendly elliptic curve BLS12-381 [Bo17] for 128-bit security, **with point compression**, $\mathbb{G}_1$ points can be represented with 48 bytes and $\mathbb{G}_2$ points with 96 bytes. Since our initial ciphertexts are trivially encrypted with fixed randomness and Adapt does not require the verification key as input, only the signature needs to be included in the QR code of each slip. At this level of security, the malleable signature requires $3 * 48 + 96 = 240$ bytes, keeping the QR code compact. Even if we add the verification key explicitly in the QR code, meaning that the booth devices BD do not even need to know the list of verification keys, we only need 432 bytes plus the candidate name. With the highest level of error correction this would fit a QR code version 23 which is 109×109 in size.

4 Protocol Description

For simplicity of presentation, we assume a simple voting method: voters choose one of n possible candidates. Generalizations to, for example, approval voting, are straightforward.

4.1 The Setup Phase

The details of the election, the candidates, the voting method, the start and end dates, the election ID, etc., are published on the Bulletin Board (BB). An Election Authority (EA) public key for encryption ek_{EA} together with signature keys for authentication of the election data are published on the BB.

N distinct malleable signature key pairs, $(\mathsf{sk}_j, \mathsf{vk}_j)$, are generated, sufficient to cover the electorate and leave some to spare. The public/verification keys vk_j are published on the BB. For each $j \in N$, sk_j is used to sign a trivial and malleable encryption for each of the n ballot options.

$$
\begin{aligned}
&\mathsf{Sign}_M(\mathsf{sk}_j, \mathsf{ek}_{EA}, \mathsf{Enc}(\mathsf{ek}_{EA}, \mathrm{Candidate}_1 \| \mathrm{ElecID}; \mathsf{r} = 0)),\\
&\mathsf{Sign}_M(\mathsf{sk}_j, \mathsf{ek}_{EA}, \mathsf{Enc}(\mathsf{ek}_{EA}, \mathrm{Candidate}_2 \| \mathrm{ElecID}; \mathsf{r} = 0)),\\
&\quad\vdots\\
&\mathsf{Sign}_M(\mathsf{sk}_j, \mathsf{ek}_{EA}, \mathsf{Enc}(\mathsf{ek}_{EA}, \mathrm{Candidate}_n \| \mathrm{ElecID}; \mathsf{r} = 0)).
\end{aligned}
$$

These terms will be used to generate the content of the jth paper ballot packet. Each signature over a trivial encryption of the candidate is printed on a separate slip along with vk_j, and these slips will be placed in a tamper evident envelope. Thus, the ith slip in the jth envelope will thus show:

$$\mathsf{vk}_j,\ \mathsf{H}(\mathsf{vk}_j),\ \mathsf{Sign}_M(\mathsf{sk}_j, \mathsf{ek}_{EA}, \mathsf{Enc}(\mathsf{ek}_{EA}, \text{Candidate}_i \| \text{ElecID}; \mathsf{r} = 0))$$

Where $\text{Candidate}_i \| \text{ElecID}$ denotes the ith candidate concatenated with the election ID. Sign_M denotes the malleable signature and $\mathsf{Enc}(\mathsf{ek}, \mathrm{X}; \mathsf{r})$ denotes the re-randomisable encryption under the public key ek of X with randomisation r. As the El Gamal encryption of Candidate_i has trivial randomization, it shows the candidate in the clear. What is printed on the slip as just the malleable signature along with the candidate name (or identifier) in the clear. This a typical slip in one of the ballot packs might look symbolically like this:

$$\begin{array}{c} \text{Candidate}_i \\ \mathsf{H}(\mathsf{vk}_j) \\ \text{QR-Code}_{(i,j)} = \mathsf{Sign}(\mathsf{sk}_j, \mathsf{ek}_{EA}, \mathsf{Enc}(\mathsf{ek}_{EA}, \text{Candidate}_i; 0)) \end{array}$$

where Sign_M is the malleable signature algorithm on randomisable ciphertexts, Enc is the re-randomisable public key encryption algorithm, $(\mathsf{sk}_j, \mathsf{vk}_j)$ denotes the signing and verification key pair for the jth pack, ek_{EA} is the public encryption key, and $\mathsf{r} = 0$ specifies the fixed random coins used during encryption. $\mathsf{H}(\mathsf{vk}_j)$ denotes a short hash of the verification key. This can be very short; it suffices that the verification keys map to distinct values, to allow the voter to uniquely identify their ballot online.

To the voter, a ballot sheet might look like this, where the QR code encodes the malleable signature over the trivial encryption of the candidate:

4.2 Auditing the setup of the BB

Auditors should verify that the verification keys vk_j and their hash fingerprints associated with the packs are pairwise distinct and correctly computed.

4.3 Auditing the Paper ballot packs

We run a pre-audit of the printed packs to identify any problems in the setup early. For this auditors open up a random selection of the envelopes and check the contents for well-formedness: that each envelope contains N slips, each baring a valid signature over one of the candidates. For the jth pack, each signature should validate against vk_j and all candidates should appear in the pack.

Envelopes that are audited at this stage should be set aside and not used for casting a vote, in order to avoid compromising the secrecy of the signatures during the voting period. Any leftover packs after voting has closed will be opened and audited.

We want to audit the paper packs that are used to cast votes, thus achieving a full, public audit of all ballots. This can be achieved by requiring voters to drop all slips, including the scanned one, into an "audit box" in the booth. Once voting has closed, the content of these can be posted to the BB and can be universally audited.

4.4 Runtime audits

The pre-audits described above will be effective at detecting ill-formed packs in the initial batch. However, we do not want to rely on chain-of-custody assumptions to ensure that no packs are corrupted after these audits but before use. In addition to the audits of the setup described above, independent auditors perform random spot checks of the ballot packs throughout the voting period. Furthermore, when a voter is presented with a pack, they can opt to have voter assistants run validity checks on the signatures. Auditing apps could be made available at polling stations to allow voters to perform their own checks on the signatures. As part of the voting procedure described below, officials should also check the validity of the signature on the printout that the voter brings back from the booth. The validity of this (transformed) signature implies that the original signature on the paper slip was also valid.

4.5 The Voting Ceremony

After registering at the polling station, the voter is assigned, or chooses, a ballot pack at random. The voter then goes to the booth and selects and scans the sheet displaying the

candidate of choice. Ideally, for higher assurance, the voters could apply their own malleable signature, but this, of course, requires additional equipment and infrastructure, and an extra step in the voting ceremony with potential coercion threats. We leave this for future work. Officials can note a ballot pack tag to prevent attacks such as chain voting.[5]

The Booth Device (BD) (re-)encrypts the chosen candidate and adapts the signature in line with the randomisation. The BD also generates a ZK proof of knowledge of the randomization it used for (re)encryption. The plaintext showing the candidate is no longer visible on the printout due to the re-encryption of the original, trivial encryption.

The voter drops all N of the original slips in a "audit box", leaves the booth with the printout to the registration desk where the officials check the validity of the transformed signature. Assuming that these checks go through, the ballot is scanned, and posted to the BB. Later, all posted signatures will be universally verified. The receipt, a copy of the posted ballot, is then franked, digitally signed, and handed back to the voter. An additional copy of the receipt can be kept locally as a backup to verify the consistency with what is posted to BB. The fact that a ballot with verification key vk_j was cast is logged.

Optionally, a paper audit trail could be created by, for example, adding a VVPAT mechanism: the BD prints the candidate onto an additional sheet in the booth, which the voter checks, and it is then dropped into a ballot box (distinct from the audit box).

4.6 The Voters' Verification Steps

On receiving a ballot pack, the voter should check that the envelope is sealed with no evidence of tampering. In the booth, they should take care to present only the slip showing their candidate to the scanner. The voter should also check that the signature verification key vk_j shown on the printout matches that on their pack. This will also be checked by the officials, so this voter check is less critical.

Once the vote has closed, voters should check that their receipt appears correctly on the BB. This can be backed up by checks performed by independent auditors based on local copies of the cast ballots.

5 Analysis of the In-person Scheme

The security and privacy of schemes using malleable signatures or TREnc have been thoroughly analysed both with pen-and-paper proofs and formally [CFL19; DPP22; HG21]. Here, we focus on the key novel ingredients.

[5] While standard chain voting, where the adversary directly marks the preferred candidate, is not possible, the coercer could instead invalidate the QR codes of all other choices but the preferred.

Cast-as-intended. The key element of the argument for cast-as-intended is the claim that the only printout that the booth device can produce with a valid signature for the jth pack is over an encryption of the vote chosen by the voter. For this to hold we need to ensure:

- The BD has no access to signing keys.
- The only term bearing a valid signature for jth pack that the BD has access to is the one presented to the scanner by the voter.
- The signatures cannot forged, i.e. can only be generated either with knowledge of the signing key or by transformation of a valid signature.

The first follows from standard protection measures ensuring the secrecy of the signature keys. Note that the signing keys can and should be deleted as soon as the setup steps are completed.

The second assumption will follow from the chain-of-custody measures around the ballot packs. This will include good tamper-evidence seals on the envelopes, but officials and voters should check that such seals are intact when packs are handed out. We need also to assume that the signing authority does not leak any signatures. We do have additional protection in depth: the isolation of the BSs should ensure that even if keys or signatures are leaked, an attacker should not be able to infiltrate these into the BDs. Of course, the isolation and minimal functionality of the BDs have to be enforced for this to be effective.

It is important that the voter presents only the correct slip to the device. If these assumptions hold then the only valid signature under the pack's verification key vk that the device can compute and print will be over a re-encryption of the voter's choice.

The dependency on keeping the (malleable) signatures secret is reminiscent of code-based voting, where cast-as-intended toward the voter's device also relies on it not knowing the codes.

We stress again that the CaI property provided by our scheme is conditional on the assumptions stated above. Many definitions of CaI (or E2E V) require it to be unconditional, i.e. not dependent on any assumptions, and so our scheme provides a weaker notion. We believe however that these assumptions are reasonable and that this weakening of the property is fair trade-off against the benefits. We note further that voters still have the option of performing an audit of the output of the BD so we still have an underpinning of the usual unconditional E2E verifiability. [6]

Privacy. The booth devices BD are trusted separately for privacy; however, we note that a malicious device would not be able to break the privacy of voters using honest devices. The

[6] If a voter does perform such an audit, then a fresh pack should be used to create a new output, to ensure that the BD only ever sees one signature per pack.

ZK proof of knowledge of the randomisation factors created by the BDs render the output ciphertext (labelled) IND-CPA and hence prevents ballot copy attacks.

Receipt-Freeness. Receipt-freeness relies on the secrecy of the randomness used by the booth devices to randomize the ciphertexts. As mentioned in Section 3 the output rerandomised ciphertext will be indistinguishable from a ciphertext of another candidate using the class-hiding and signature-adaptation security properties.

Caveat: Robustness of Public Key Schemes. Several works have pointed out shortcomings of public key definitions when considering multiple public keys which are not captured by standard security definitions [GNR19]. In particular, the same signature might verify against several (maliciously generated) verification keys [Ja19]. In our case, verifiability attacks using such cross-signatures are prevented by the assumption that the offline devices BD do not know any other signatures. Privacy attacks could be more troublesome, but we note that if a signature on a ballot sheet with vk_j is crafted to also verify against another $\mathsf{vk}_{j'} \neq \mathsf{vk}_j$, a re-randomization which is not controlled by the adversary should, with overwhelming probability, no longer verify against $\mathsf{vk}_{j'}$ for most schemes. In our case, the ZK proof, mentioned above, also prevents ballot copy attacks.

6 Conclusions

We have presented an approach to achieving E2E-V that does not require voters to perform ballot audits, rather these are performed by observers. This makes the voting experience a simple vote an go and ensures that elections are not only verifiable, but actually verified. We thus have evidence-based schemes that are much closer to conventional voting, but provide greater assurance. This means shifting the basis of trust away from the voters to those of auditors, but this should be an acceptable trade-off in many contexts.

In our scheme, all ballot forms, including the ballots that are cast, are independently audited. The need for voters to perform their own ballot audits is avoided; they need only to check that the correct candidate appears on the paper slip that they present to the booth device. Furthermore, ballot auditing now takes the form of a simple signature verification, as opposed to the usual verification of an encryption. This brings several advantages:

1. Cast-as-intended assurance is vastly higher than that achievable using cut-and-choose techniques (where cast ballots are not themselves audited).
2. Voters are not expected to perform ballot audits themselves. This makes the approach far more usable and accessible than traditional cut-and-choose techniques.[7]

[7] However, voters can be offered the option of performing their own audits if they wish.

3. The fact that all ballots are audited ensures that the CaI property is verified, rather than just verifiable.
4. Ballot auditing does not impinge on vote privacy.
5. Dispute resolution for ballot audits is automatic: ballots are either well-formed or not.

We argue that it is possible to outsource the cast as intended checks usually considered as individual, i.e. necessarily performed by the voters. This comes at the cost of introducing trust assumptions and loses the unconditional nature that many E2E V schemes provide, but we believe that the benefits outweigh the downsides, at least in some contexts. We have presented an in-person scheme that realises these goals.

It is complex to perform universal auditing of all the ballots **before** the election without undermining the verifiability by revealing signatures (but see item 4 below). However, the fact that we perform a full audit of all the ballots, including cast ballots after voting has closed, by auditing the contents of the audit boxes, does provide a form of universal ballot auditing.

For future work, we plan to investigate:

1. Complete a rigorous the analysis of the scheme.
2. Incorporating voter's (malleable) signatures.
3. Incorporating a paper audit trail and RLA techniques.
4. Committing the ballot pack information to the BB under encryption and performing universal pre-auditing using ZK proofs of existence of signatures under encryption.
5. Develop online variants of the approach.

References

[Be06] Benaloh, J.: Simple Verifiable Elections. In: 2006 Electronic Voting Technology Workshop, EVT '06. USENIX Association, 2006.

[BF20] Bauer, B.; Fuchsbauer, G.: Efficient Signatures on Randomizable Ciphertexts. In (Galdi, C.; Kolesnikov, V., eds.): Security and Cryptography for Networks. Springer International Publishing, pp. 359–381, 2020.

[Bl11] Blazy, O. et al.: Signatures on Randomizable Ciphertexts. In: PKC 2011. Springer, pp. 403–422, 2011.

[Bo17] Bowe, S.: BLS12-381: New zk-SNARK elliptic curve construction, https://electriccoin.co/blog/new-snark-curve/, 2017.

[BPW12] Bernhard, D.; Pereira, O.; Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp. 626–643, 2012.

[CFL19] Cortier, V.; Filipiak, A.; Lallemand, J.: BeleniosVS: Secrecy and verifiability against a corrupted voting device. In: IEEE CSF 2019. Pp. 367–36714, 2019.

[CGG19] Cortier, V.; Gaudry, P.; Glondu, S.: Belenios: a simple private and verifiable electronic voting system. In: Foundations of Security, Protocols, and Equational Reasoning. Springer, pp. 214–238, 2019.

[Ch16] Chaidos, P. et al.: BeleniosRF: A non-interactive receipt-free electronic voting scheme. In: ACM CCS 2016. Pp. 1614–1625, 2016.

[CM17] Clarke, D.; Martens, T.: Real-world Electronic Voting: Design, Analysis and Deployment. In. CRC Press, chap. E-Voting in Estonia, pp. 129–141, 2017.

[Da24] Damodaran, A. et al.: Hyperion: Transparent End-to-End Verifiable Voting with Coercion Mitigation, Cryptology ePrint Archive, Paper 2024/1182, 2024, https://eprint.iacr.org/2024/1182.

[DPP22] Devillez, H.; Pereira, O.; Peters, T.: Traceable receipt-free encryption. In: ASIACRYPT 2022. Springer, pp. 273–303, 2022.

[FHS14] Fuchsbauer, G.; Hanser, C.; Slamanig, D.: Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials, Cryptology ePrint Archive, Report 2014/944, https://eprint.iacr.org/2014/944, 2014.

[GNR19] Géraud, R.; Naccache, D.; Roşie, R.: Robust encryption, extended. In: Topics in Cryptology–CT-RSA 2019, Proceedings. Springer, pp. 149–168, 2019.

[HG21] Haines, T.; Goré, R.: Improved Verifiability for BeleniosVS. Cryptology ePrint Archive, 2021.

[HPP20] Hébant, C.; Phan, D. H.; Pointcheval, D.: Linearly-Homomorphic Signatures and Scalable Mix-Nets. In: Public-Key Cryptography – PKC 2020. Springer International Publishing, Cham, pp. 597–627, 2020.

[HS14] Hanser, C.; Slamanig, D.: Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials. In (Sarkar, P.; Iwata, T., eds.): Advances in Cryptology – ASIACRYPT 2014. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 491–511, 2014.

[Ja19] Jackson, D. et al.: Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. Pp. 2165–2180, 2019.

[Ka11] Karayumak, F. et al.: Usability Analysis of Helios - An Open Source Verifiable Remote Electronic Voting System. In: 2011 EVT/WOTE '11. USENIX Association, 2011.

[Mo24] Mosaheb, R. et al.: Direct and Transparent Voter Verification with Everlasting Receipt-Freeness. In: International Joint Conference on Electronic Voting. Springer Nature Switzerland Cham, pp. 124–140, 2024.

[Ne04] Neff, C. A.: Practical High Certainty Intent Verification for Encrypted Votes, Unpublished manuscript, 2004.

[RRI16] Ryan, P. Y. A.; Rønne, P. B.; Iovino, V.: Selene: Voting with transparent verifiability and coercion-mitigation. In: International Conference on Financial Cryptography and Data Security. Springer, pp. 176–192, 2016.

[Ry09] Ryan, P. Y. A. et al.: Prêt à Voter: a Voter-Verifiable Voting System. IEEE Transactions on Information Forensics and Security 4 (4), pp. 662–673, 2009.