OpenVoting: Recoverability from Failures in Dual Voting

Prashant Agrawal[1] Kabir Tomer[2*] Abhinav Nakarmi[3] Mahabir Prasad Jhanwar[3]
Subodh Sharma[1] Subhashis Banerjee[1,3]

[1]Department of Computer Science and Engineering, IIT Delhi, {prashant,svs,suban}@cse.iitd.ac.in
[2]Department of Computer Science, University of Illinois Urbana-Champaign, ktomer2@illinois.edu.in
[3]Department of Computer Science and Centre for Digitalisation, AI and Society, Ashoka University,
abhinav.nakarmi@alumni.ashoka.edu.in, {mahavir.jhawar,suban}@ashoka.edu.in

Abstract

In this paper we address the problem of recovery from failures without re-running entire elections when elections fail to verify. We consider the setting of *dual voting* protocols, where the cryptographic guarantees of end-to-end verifiable voting (E2E-V) are combined with the simplicity of audit using voter-verified paper records (VVPR). We first consider the design requirements of such a system and then suggest a protocol called *OpenVoting*, which identifies a verifiable subset of error-free votes consistent with the VVPRs, and the polling booths corresponding to the votes that fail to verify with possible reasons for the failures.

1 Introduction

Conducting large-scale public elections in a dispute-free manner is not an easy task. On the one hand, there are end-to-end verifiable voting (E2E-V) systems [1, 11, 24, 8, 4] that provide cryptographic guarantees of correctness. Although the guarantees are sound, these systems are not yet very popular in large public elections. As the German Constitutional Court observes [18], depending solely on cryptographic guarantees is somewhat untenable as verification of election results requires expert knowledge. Moreover, in case voter checks or universal verifications fail, the E2E-V systems do not provide easy methods of recovery without necessitating complete re-election [6].

On the other hand, there are systems that rely on paper-audit trails to verify electronic tallies [25, 21, 16]. These systems maintain reliable records of cleartext voter-marked paper ballots or voter-verified paper records (VVPRs) alongside electronic vote records. They use electronic counting for efficiency and conduct easy-to-understand statistical audits, called *risk-limiting audits* (RLAs), to demonstrate that the electronic winners match the winners that would be declared by a full paper count. In case of conflict, the electronic outcome is suggested to be replaced by the paper one. However, these systems require the electorate to trust that the paper records correctly represent voter intent and are not corrupted in the custody chain from the time of voting to that of counting or auditing.

Dual voting approaches, where the voting protocols support simultaneous voting for both the cryptographic and the VVPR-based systems [5, 13, 22, 17, 4, 12], combine the cryptographic guarantees of E2E-V systems with the simplicity and adoptability of paper records. However, in most existing dual voting systems, one typically ends up running two parallel and independent elections, only coupled loosely through simultaneous voting for both in the polling booth. If the electronic and paper record systems are not tightly coupled, and demonstrably in one-to-one correspondence, then it begs the questions: which ought be the legal definition of the vote, and, in case of a tally mismatch, which should be trusted? Why? And how to recover from errors?

It appears that existing approaches either do not provide any recovery mechanism or recover by privileging VVPR counts over electronic counts. In large public elections running simultaneously at multiple polling booths per constituency, failures due to intended or unintended errors by different actors are expected. Polling officers may upload wrong encrypted votes, backend servers may decrypt votes incorrectly, paper records may be tampered with during the custody chain, and voters may put bogus votes in ballot boxes to discredit the election. Discarding the entire election due to failures caused by some bad actors or completely trusting the VVPRs are both unsatisfactory solutions.

*Work done while at IIT Delhi.

In this paper, we study the problem of *recoverability* of a dual voting protocol from audit failures. We consider large, multi-polling booth, first-past-the-post elections like the national elections in India. We observe that except for backend failures, most of the other failures are due to localised corruption of individual polling booths. Therefore, we propose to identify the offending polling booths and perform a local re-election — if at all required — only at those polling booths. Errors — despite the best efforts to minimise them — are inevitable in large elections and such localised recovery may considerably improve the election's overall robustness and transparency.

However, recoverability has a natural tradeoff with vote secrecy. For example, a naive approach that simply publishes and audits votes for each polling booth reveals voting statistics of each booth. In electoral contexts where voters are assigned a specific polling booth according to their residential neighbourhoods, with only a few thousand voters per booth, e.g., in India, revealing booth-level voting statistics poses a significant risk of localised targeting and coercion [3]. Our approach minimises booth-level voting data exposure, disclosing only what is absolutely necessary for recovery.

Main contributions. 1) We analyse the design requirements for a recoverable and secrecy-preserving dual voting protocol (Section 2). **2)** We formalise the notion of recoverability and secrecy in terms of the capability to verifiably identify polling booths contributing to verification failures and extract a verifiable subset of error-free votes in zero-knowledge (Section 3). **3)** We propose a novel dual-voting protocol called *OpenVoting* that satisfies our notions of recoverability and secrecy (Section 4).

Related work. Dual voting was introduced by Benaloh [5], following which multiple dual voting protocols emerged [17, 13, 22, 4, 12]. Bernhard et al. [6] gives a comprehensive survey of the tradeoffs and open problems in E2E-V and RLA-based voting.

Rivest [23] proposed the notion of *strong software independence* that is similar to our notion of recoverability. It demands that a detected change or error in an election outcome (due to a change or error in the software) can be corrected without re-running the (entire) election. However, "correcting" errors without re-running even parts of an election requires a ground truth, which is usually assumed to be the paper audit trail. Instead, we propose partial recoverability via fault localisation, without completely trusting either paper or electronic votes. The notion of *accountability* [15] is also related, but it is focused on assigning blame for failures and not on recovering from them.

2 Design Requirements

In a typical dual voting protocol, the vote casting process produces a) a VVPR containing the voter's vote in cleartext and b) a voter receipt containing an encryption of the vote. The encrypted votes are published on a bulletin board, typically by a *polling officer*, and are processed by a cryptographic *backend* to produce the electronic tally. The backend typically consists of multiple independent servers which jointly compute the tally from the encrypted inputs, provide a proof of correctness, and preserve vote secrecy unless a threshold number of servers are corrupted. VVPRs counted together produce the paper tally.

Our high-level goal is to publicly verify whether both tallies represent true voter intents and whether all public outputs are consistent with each other. If not, the aim of recovery is to identify booths contributing to the inconsistencies, and segregate the outputs produced by other error-free booths, without leaking any additional information. For this, the protocol design must fundamentally have the following features:

1. The backend must publish individual decrypted votes with matching identifiers with the VVPRs[1], to narrow down tally inconsistencies to individual vote mismatches.

2. The encrypted votes must have voter and booth identifiers. The former enable matching with voter receipts; the latter enable identifying booths in case of errors.

3. The decrypted votes and VVPRs and their identifiers must be unlinkable to encrypted votes, voter receipts or voter identifiers to ensure vote secrecy. They should also be unlinkable to the booth identifiers to hide booth-level voting statistics.

4. For the same reason, VVPRs should be revealed and counted only after aggregating them over all the polling booths.

[1] Homomorphic tallying based backends [1, 4] report only the final tally and do not support this.

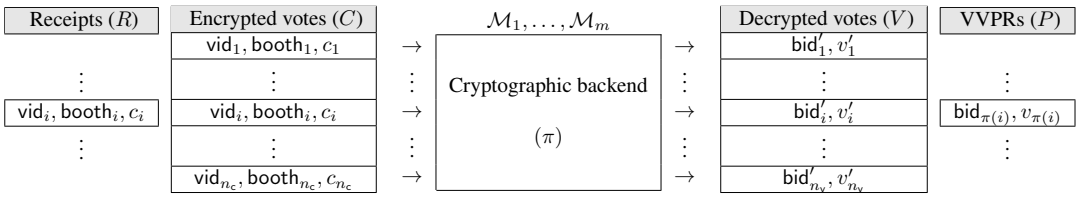

Figure 1: A recoverable dual voting protocol design. The VVPR for a voter with identifier vid_i voting at booth booth_i contains a ballot identifier bid_i and cleartext vote v_i. Her encrypted vote c_i encrypts a value, e.g., (bid_i, v_i), that when decrypted can be uniquely matched with the corresponding VVPR. Decrypted votes are published by backend servers $\mathcal{M}_1, \ldots, \mathcal{M}_m$ in a permuted order under a secret shared permutation π such that $(\mathsf{bid}'_i, v'_i) = (\mathsf{bid}_{\pi(i)}, v_{\pi(i)})$. Note that n_c and n_v denote the number of encrypted votes and decrypted votes respectively.

Input-phase failures[1,2]	
FI_1	A receipt r against vid exists in R but no encrypted vote against vid exists in C
FI_2	The encrypted vote c in C against vid does not match the receipt r in R against vid
Mixing-phase failures[2,3]	
FM_1	An encrypted vote c in C does not decrypt to any cleartext vote (bid, v) in V
FM_2	A cleartext vote (bid, v) in V is not obtained by decrypting any encrypted vote c in C
FM_3	Two encrypted votes in C decrypt to the same cleartext vote (bid, v) in V
Output-phase failures[2]	
FO_1	An (electronic) decrypted vote (bid, v) exists in V but no VVPR against bid exists in P
FO_2	A VVPR (bid, v) exists in P but no decrypted vote against bid exists in V
FO_3	The decrypted vote v against bid in V does not match the cleartext vote in the VVPR against bid in P
FO_4	Two decrypted votes in V match with a single VVPR (bid, v) in P
FO_5	Two VVPRs in P match with a single decrypted vote (bid, v) in V
Cast-as-intended failures	
FC	A receipt r obtained at a polling booth j does not encrypt the voter's intended vote correctly

[1] A spurious encrypted vote against a vid in C without a receipt in R against that vid is not considered a failure, because some voters may not upload their receipts. Also, we do not consider duplicated receipts and encrypted votes because vids are assumed to be unique identifiers.
[2] We only consider authentic entries in R, C, V and P. Failures where the authenticity of these items cannot be verified are considered equivalent to failures where they are not even uploaded. Receipts and VVPRs are authenticated by official stamps and encrypted and decrypted votes by appropriate digital signatures.
[3] The case of a single encrypted vote in C decrypting to two different entries in V is not considered because this will result in duplicated entries in V, which can be clearly attributed to backend failures and removed without any dispute.

Figure 2: Potential failures given public outputs (R, C, V, P).

The encrypted and decrypted votes must be published on two public bulletin boards to enable voters to match their receipts and public verification of the electronic tally. It will also be helpful to upload all VVPRs after scanning, and as many voter receipts as possible, to two other bulletin boards for better transparency and public verifiability. We depict such a design in Figure 1.

Note that the public outputs in Figure 1 are effectively *claims* endorsed by various entities as to what should be the correct vote: receipts by voters, encrypted votes by polling officers, decrypted votes by the backend servers, and VVPRs by the VVPR counting authorities. We group disputes between these claims into *input-phase failures*, for mismatches between published voter receipts and encrypted votes, *mixing-phase failures*, for mismatches between encrypted votes and decrypted votes, and *output-phase failures*, for mismatches between decrypted votes and VVPRs (see Figure 2). Further, we categorise claims of receipts not encrypting voter intents correctly as *cast-as-intended failures*. Given these failures, recoverability requires an audit protocol that verifies whether the different claims for a given vote are consistent, resolves disputes otherwise, and narrows down the affected votes when the disputes are unresolvable.

To recover from input-phase failures, it is not sufficient if a statistically significant sample of voters from the entire constituency verify their receipts, because in case of any failure, all the uploaded encrypted votes become untrustworthy. Thus, the population for sampling must be each polling booth. This does increase the voter verification overhead, but offers better localisation of errors and recovery.

Recoverability from mixing-phase failures requires that in case the output list of decrypted votes is not correct, individual failing entries — encrypted votes whose decryptions were not available in V and individual decrypted votes that were not decrypted by any encrypted vote on C — should be verifiably identified by the backend servers. And, this must be achieved without leaking any additional information.

Recoverability from output-phase failures requires identifying which of the electronic vote and the VVPR represents the voter's intent. This may be possible in some cases but not always. For example, if the voter's receipt is available on R, then the dispute can be resolved if one can verify in zero-knowledge that the receipt encrypted the electronic vote and not the paper one, or vice versa.

In some cases, the disputes may not be resolvable at all. Consider case FO_3 in Figure 2 and suppose the receipt is not available. FO_3 may be due to a) the polling officer uploading an encrypted vote not matching the voter's receipt; b) the voter dropping a bogus VVPR into the ballot box; c) a malicious agent altering the VVPRs post-polling; or d) the backend servers not decrypting the uploaded encrypted vote correctly. Different cases point to failures in either the electronic vote or the VVPR and it is not possible to identify the true voter intent. Thus, a conservative way to recover from this situation is to identify the polling booth where the dispute may have originated and conduct only a local re-election at this booth. This must be done without revealing polling booth statistics of at least the uncorrupted polling booths.

The required action in all the above cases can be reduced to the backend proving in zero-knowledge that an encrypted vote corresponds to one of a set of decrypted votes (a *distributed ZKP of set-membership* [2]), or that a clear-text vote is a decryption of one of a set of encrypted votes (a *distributed ZKP of reverse set-membership* [2]).

Cast-as-intended failures may typically happen in two ways. First, ballots may be malformed. Protection against this threat requires a separate audit of a statistically significant sample of ballots before vote casting. Recoverability additionally requires ballot audits to be performed per polling booth. Second, ballots or receipts may be marked incorrectly. In dual voting systems based on hand-marked ballots, the voter may mark the encrypted and the VVPR parts differently, leading to failures. Although this is easily detected and invalidated during VVPR audit, fixing accountability may be difficult and hence voters may do this deliberately to discredit the election. In systems based on ballot marking devices (BMD), such voter errors are avoided but a dispute may be raised that the ballot marking is not according to the voter's choice. Such a dispute between a man and a machine is unresolvable and the only recourse is to allow the voter to revote. This may however cause a deadlock, which can only be resolved through a social process. Still, a BMD should be a preferred option for dual voting since it minimises voter-initiated errors.

3 Formalisation

We now formalise the requirements outlined in the previous section. Given a positive integer x, let $[x]$ denote the set $\{1, \ldots, x\}$. We consider a dual-voting protocol involving α candidates, n voters $(\mathcal{V}_i)_{i\in[n]}$, τ ballot generators $(\mathcal{G}_t)_{t\in[\tau]}$, ℓ polling booths consisting of polling officers $(\mathcal{P}_j)_{j\in[\ell]}$, BMDs $(\mathcal{D}_j)_{j\in[\ell]}$ and physical ballot boxes $(\mathcal{B}_j)_{j\in[\ell]}$, m backend servers $(\mathcal{M}_k)_{k\in[m]}$, and an auditor $\mathcal{A}$. We also assume existence of a public bulletin board where lists R, C, V and P are published. We consider a protocol structure (Setup, $\mathsf{BallotGen}$, Cast, Tally, $\mathsf{BallotAudit}$, $\mathsf{ReceiptAudit}$, $\mathsf{TallyAudit}$) where:

- Setup is a protocol involving $(\mathcal{G}_t)_{t\in[\tau]}$, $(\mathcal{P}_j)_{j\in[\ell]}$ and $(\mathcal{M}_k)_{k\in[m]}$ to generate public/private key pairs and other public election parameters.
- $\mathsf{BallotGen}$ is a protocol involving $(\mathcal{G}_t)_{t\in[\tau]}$ to securely print a sealed ballot given a booth identifier $j \in [\ell]$.
- Cast is the vote casting protocol involving $\mathcal{V}_i$ and $\mathcal{P}_j$, $\mathcal{D}_j$, $\mathcal{B}_j$ at booth $j \in [\ell]$ assigned to $\mathcal{V}_i$. $\mathcal{V}_i$'s input is its intended vote v and a ballot b. The protocol outputs a voter receipt r, an encrypted vote c and a VVPR p such that p gets dropped in ballot box $\mathcal{B}_j$, $\mathcal{V}_i$ takes r home and $\mathcal{P}_j$ uploads c on C. The voter may or may not publish r on R. The VVPR is published on P after aggregating VVPRs from all the booths.
- Tally is the vote processing/tallying protocol involving $(\mathcal{M}_k)_{k\in[m]}$ where they take as input the encrypted votes $(c_i)_{i\in[n]}$ published on C, permute and decrypt them and publish a list $(v'_i)_{i\in[n]}$ of decrypted votes on V.
- $\mathsf{BallotAudit}$ is a protocol involving $\mathcal{A}$ and $\mathcal{P}_j$ executed at each booth j to verify if ballots at booth j are well-formed.
- $\mathsf{ReceiptAudit}$ is a protocol involving $\mathcal{A}$ and the voters to verify that voter receipts at booth j match those uploaded on list C.
- $\mathsf{TallyAudit}$ is a protocol involving $\mathcal{A}$, $(\mathcal{M}_k)_{k\in[m]}$ and $(\mathcal{G}_t)_{t\in[\tau]}$ to verify whether the electronic and paper tallies are correct and narrow down errors if not. It takes as input all published lists (R, C, V, P) and lets $\mathcal{A}$ output a tuple (J^*, V^*) where J^* denotes the set of booths that contributed potentially outcome-changing failures and

V^* denotes the set of votes from booths not in J^* ($\mathcal{A}$ may also be aborted). The expected usage of the (J^*, V^*) output is that in case of failures/disputes, the election could be rerun at the booths in J^* and the rerun results could be merged with the recovered partial tally from V^* to obtain the complete election tally. Results are announced to the general public only after TallyAudit has finished.

Note that although the above audits are performed by different auditors (even voters) at different times and places, we simplify by representing all the auditors by $\mathcal{A}$.

Let ϵ_{b} denote the probability that BallotAudit passes at some booth j yet a receipt from the booth does not encrypt the voter's intent correctly, and ϵ_{r} denote the probability that ReceiptAudit passes for booth j yet a receipt from the booth is not uploaded correctly. Further, let $R^* \subseteq R$, $C^* \subseteq C$ and $P^* \subseteq P$ respectively denote receipts, encrypted votes and VVPRs from booths not in J^*. Finally, let *failures* in a tuple (R, C, V, P) be as defined in Figure 2 with the added condition that if a receipt or encrypted vote from a booth fails with input-phase or cast-as-intended failures, then all receipts and encrypted votes from that booth are considered as failures.

Definition 1 models our notion of recoverability parametrised by probabilities ϵ_{b} and ϵ_{r} denoting the effectiveness of ballot and receipt audits. The case when J^* is empty denotes that no rerun is required at any booth, either because the election ran completely correctly, or because the number of failures are small compared to the reported winning margin. When non-empty, J^* should exactly be the set of booths where re-run is required because of failures that may affect the final outcome and votes V^* must be consistent with receipts, encrypted votes and VVPRs from booths not in J^*.

Note that the auditor is allowed to abort the TallyAudit protocol, since if the mix-servers and the ballot generators holding the election secrets do not cooperate, then recovery cannot happen. This is not an issue because unlike polling booth failures, these failures are centralised and non-cooperation directly puts the blame on these entities.

Definition 1 (Recoverability)**.** *A voting protocol* (Setup, BallotGen, Cast, Tally, BallotAudit, ReceiptAudit, TallyAudit) *is* recoverable *by the audit protocols if for all polynomially bounded adversaries corrupting* $(\mathcal{G}_t)_{t\in[\tau]}$, $(\mathcal{M}_k)_{k\in[m]}$, $(\mathcal{P}_j)_{j\in[\ell]}$, $(\mathcal{D}_j)_{j\in[\ell]}$ *and* $(\mathcal{B}_j)_{j\in[\ell]}$ *such that* $\mathcal{A}$ *outputs a tuple* (J^*, V^*) *and does not abort, the following conditions hold true with probability only negligibly smaller than* $1 - \ell(\epsilon_{\mathsf{b}} + \epsilon_{\mathsf{r}})$*:*

- *if* J^* *is empty, then the number of failures in* (R^*, C^*, V^*, P^*) *is less than the reported winning margin computed from* V*; and*
- *if* J^* *is non-empty, then* (R^*, C^*, V^*, P^*) *does not contain any failures and* J^* *is exactly the set of booths that contributed some failing receipt in* R*, some failing encrypted vote in* C*, or some failing VVPR in* P*.*

Definition 2 models that in the presence of the TallyAudit protocol, the standard vote secrecy guarantee is maintained except that polling booth statistics of the booths contributing some failing items are revealed. This is generally an unavoidable tradeoff.

Definition 2 (Vote Secrecy with Recoverability)**.** *A voting protocol* (Setup, BallotGen, Cast, Tally, BallotAudit, ReceiptAudit, TallyAudit) protects vote secrecy with recoverability *if no polynomially bounded adversary controlling the auditor* $\mathcal{A}$*,* $(\mathcal{P}_j)_{j\in[\ell]}$*,* $(\mathcal{D}_j)_{j\in[\ell]}$*,* $(\mathcal{G}_t)_{t\in[\tau]\setminus\{t^*\}}$ *for some* $t^* \in [\tau]$*,* $(\mathcal{M}_k)_{k\in[m]\setminus\{k^*\}}$ *for some* $k^* \in [m]$*, and* $(\mathcal{V}_i)_{i\in[n]\setminus\{i_0,i_1\}}$ *for some* $i_0, i_1 \in [n]$ *can distinguish between the following two worlds except with negligible probability:*

- ***(World 0)*** $\mathcal{V}_{i_0}$ *votes* v_0 *at booth* j_0 *and* $\mathcal{V}_{i_1}$ *votes* v_1 *at booth* j_1*, and*
- ***(World 1)*** $\mathcal{V}_{i_0}$ *votes* v_1 *at booth* j_0 *and* $\mathcal{V}_{i_1}$ *votes* v_0 *at booth* j_1*,*

where v_0, v_1 *are any two valid votes and for each failure from booth* j_0*, the adversary must create an identical failure (same failure type and affected vote) from booth* j_1*.*

4 The *OpenVoting* Protocol

4.1 Preliminaries

Notation. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ denote cyclic groups of prime order q ($q \gg \alpha, m, n, \ell$) such that they admit an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We assume that the n-Strong Diffie Hellman (SDH) assumption [7] holds in $(\mathbb{G}_1, \mathbb{G}_2)$, the decisional Diffie-Hellman (DDH) and the discrete logarithm (DL) assumptions hold in $\mathbb{G}_1$,

and that generators $g_1, h_1 \in \mathbb{G}_1$ are chosen randomly (say as the output of a hash function) so that nobody knows their mutual discrete logarithm.

Traceable Mixnets [2]. Traceable mixnets extend traditional mixnets [14] to enable the distributed ZKPs of set membership mentioned in Section 2. Thus, we use them as our cryptographic backend. In traceable mixnets, the backend servers, often also called *mix-servers*, can collectively prove answers to the following queries in zero-knowledge:

- TraceIn: whether a ciphertext c (from the mixnet's input ciphertext list) encrypts a value in a subset of output plaintexts (denoted as $(v'_i)_{i\in I'}$ for some $I' \subseteq [n]$).
- TraceOut: whether a plaintext v (from the mixnet's output plaintext list) is encrypted in one of a subset of input ciphertexts (denoted as $(c_i)_{i\in I}$ for some $I \subseteq [n]$.).

There are also batched versions of these queries called BTraceIn and BTraceOut, which prove multiple TraceIn and TraceOut queries together.

Formally, a traceable mixnet Π_{TM} is a protocol between a set of senders $S_1, \ldots, S_n$, a set of mix-servers $(\mathcal{M}_k)_{k\in[m]}$ and a querier Q and consists of algorithms/sub-protocols $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Mix}, \mathsf{BTraceIn}, \mathsf{BTraceOut})$ where:

- Keygen is a distributed key generation protocol involving $(\mathcal{M}_k)_{k\in[m]}$ that outputs a mixnet public key mpk and secret keys $\mathsf{msk}^{(k)}$ for each mix-server $\mathcal{M}_k$.
- Enc is the encryption algorithm that a sender S_i uses to create a ciphertext c_i encrypting its secret input v_i against mpk.
- Mix is the mixing protocol involving $(\mathcal{M}_k)_{k\in[m]}$ that takes as input the list of ciphertexts $(c_i)_{i\in[n]}$ uploaded by $(S_i)_{i\in[n]}$ and outputs a list of permuted plaintexts $(v'_i)_{i\in[n]}$ and a secret witness $\omega^{(k)}$ for each $\mathcal{M}_k$.
- $\mathsf{BTraceIn}$ is a protocol involving $(\mathcal{M}_k)_{k\in[m]}$ and Q that takes as input $(c_i)_{i\in[n]}$ and $(v'_i)_{i\in[n]}$ and index sets $I, I' \subseteq [n]$ (each $\mathcal{M}_k$ additionally uses $\omega^{(k)}$). At the end of the protocol, Q either outputs the subset of ciphertexts $\{c_i\}_{i\in I}$ that encrypt some plaintext in $\{v'_i\}_{i\in I'}$ or aborts.
- $\mathsf{BTraceOut}$ is a protocol involving $(\mathcal{M}_k)_{k\in[m]}$ and Q that takes exactly the same inputs as $\mathsf{BTraceIn}$. In this case, Q either outputs the subset of plaintexts $\{v'_i\}_{i\in I'}$ that are encrypted by some ciphertext in $\{c_i\}_{i\in I}$ or aborts.

The *soundness property of traceable mixnets* states that an adversary controlling all $(\mathcal{M}_k)_{k\in[m]}$ cannot make Q output an incorrect set. Their *secrecy property* states that an adversary controlling $(\mathcal{M}_k)_{k\in[m]\setminus\{k^*\}}$ for some $k^* \in [m]$, Q and $(S_i)_{i\in[n]\setminus\{i_0,i_1\}}$ for some $i_0, i_1 \in [n]$ cannot distinguish between a world where (S_{i_0}, S_{i_1}) respectively encrypt (v_0, v_1) and the world where they encrypt (v_1, v_0), if the $\mathsf{BTraceIn}$ and $\mathsf{BTraceOut}$ query outputs do not leak this information, i.e., if in all $\mathsf{BTraceIn}$ queries, $v_0 \in \{v'_i\}_{i\in I'}$ iff $v_1 \in \{v'_i\}_{i\in I'}$ and in all $\mathsf{BTraceOut}$ queries, $i_0 \in I$ iff $i_1 \in I$.

An instantiation of traceable mixnets. [2] also provides a concrete instantiation of a traceable mixnet, which we use. In this instantiation, mpk is of the form $((\mathsf{pk}_{\mathcal{M}_k})_{k\in[m]}, \mathsf{pk}_{\mathsf{EG}}, \mathsf{pk}_{\mathsf{Pa}})$, where $\mathsf{pk}_{\mathcal{M}_k}$ is the public key of any IND-CPA secure encryption scheme E, and $\mathsf{pk}_{\mathsf{EG}}$ and $\mathsf{pk}_{\mathsf{Pa}}$ are respectively public keys of $\mathsf{E}^{\mathsf{th}}_{\mathsf{EG}}$, the threshold ElGamal encryption scheme [10] with message space $\mathbb{G}_1$, and $\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}$, the threshold Paillier encryption scheme proposed in [9] with message space $\mathbb{Z}_N$ for an RSA modulus N. The secret key $\mathsf{msk}^{(k)}$ for each $\mathcal{M}_k$ consists of the secret key $\mathsf{sk}_{\mathcal{M}_k}$ corresponding to $\mathsf{pk}_{\mathcal{M}_k}$ and the k^{th} shares of the secret keys corresponding to $\mathsf{pk}_{\mathsf{EG}}$ and $\mathsf{pk}_{\mathsf{Pa}}$. Further, Enc on input a value $v \in \mathbb{Z}_q$ outputs a ciphertext of the form $(\epsilon, \gamma, (\mathsf{ev}^{(k)}, \mathsf{er}^{(k)})_{k\in[m]}, \rho_\gamma, \epsilon_r)$, where

- $\epsilon \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{Pa}}, v)$ is an encryption of v (interpreted as $v \in \mathbb{Z}_N$) under $\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}$,
- $\gamma = g_1^v h_1^r$ is a Pedersen commitment [20] to v in $\mathbb{G}_1$ under randomness $r \in \mathbb{Z}_q$,
- $\mathsf{ev}^{(k)} \leftarrow \mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, v^{(k)})$ is an encryption of a secret share $v^{(k)}$ of v,
- $\mathsf{er}^{(k)} \leftarrow \mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, r^{(k)})$ is an encryption of a secret share $r^{(k)}$ of r,

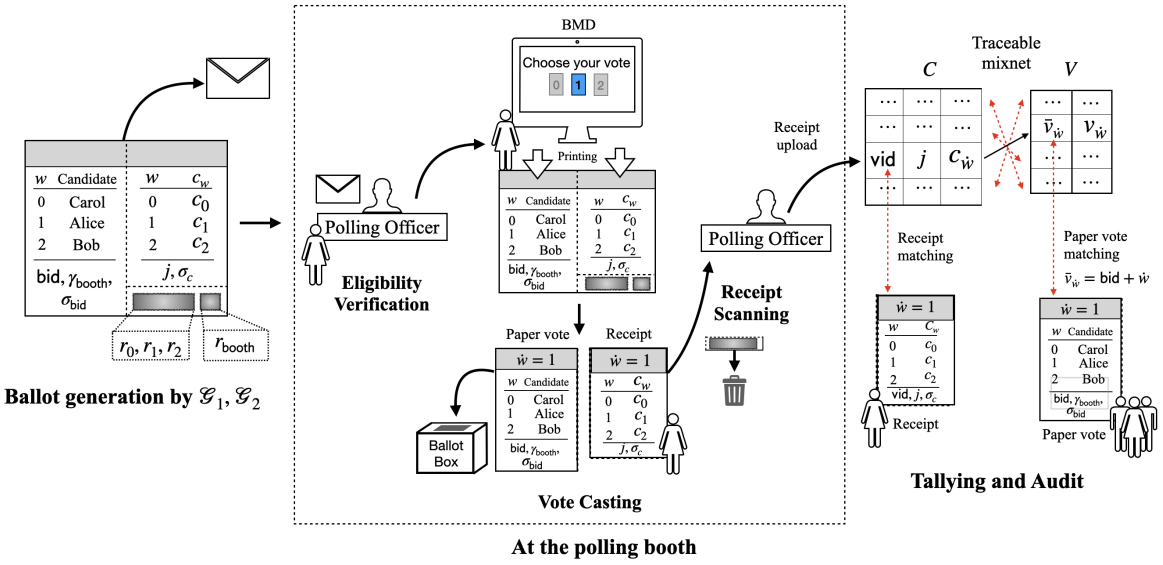

Figure 3: Overview of the *OpenVoting* protocol: w represents the candidate index in the ballot and $\dot{w}$ represents the voter's choice.

- $\rho_\gamma \leftarrow \mathsf{NIZKPK}\{(v, r) : \gamma = g_1^v h_1^r\}$ is a noninteractive ZKP of knowledge of the opening of γ, and
- $\epsilon_r \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{Pa}}, r)$ is an encryption of r (interpreted as $r \in \mathbb{Z}_N$) under $\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}$.

In our protocol, the encrypted votes are encryptions under Enc, where we instantiate scheme E with the (non-threshold) Paillier encryption scheme [19]. We need it for its following homomorphic property: given two Paillier ciphertexts c_1, c_2 encrypting messages $m_1, m_2 \in \mathbb{Z}_q$ respectively (m_1, m_2 interpreted as messages in $\mathbb{Z}_N$), the ciphertext $c_1 c_2$ encrypts the message $m_1 + m_2 \mod N = m_1 + m_2$ if $N > 2q$. We also require a public-key digital signature scheme $\Pi_{\mathsf{S}} := (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Ver})$ with the usual existential unforgeability property under chosen message attacks (EUF-CMA).

4.2 The Proposed Protocol

Figure 3 depicts the high-level *OpenVoting* protocol. Two ballot generators ($\mathcal{G}_1$ and $\mathcal{G}_2$) jointly generate sealed ballots to protect voter-vote association from both. Voters use the sealed ballots and a BMD to cast their votes. Each ballot contains two halves. The BMD prints the voter's choice on both halves without learning the vote. The left half becomes the VVPR and is deposited by the voter in a physical ballot box, while the right half becomes the voter receipt. Polling officers scan the voter receipts and upload the encrypted votes to C. The encrypted votes are processed by a traceable mixnet backend to produce decrypted votes V. Voters can verify their receipts against the encrypted votes, and VVPRs can be matched with the decrypted votes. The tally audit process uses the traceable mixnet's querying mechanism to identify polling booths contributing to failures without leaking additional information. Results are announced only after this audit step. Now we describe the sub-protocols of *OpenVoting* in detail.

4.2.1 Setup

During the Setup protocol, $\mathcal{G}_1$ and $\mathcal{G}_2$ generate public/private keys $\mathsf{pk}_{\mathcal{G}_1}, \mathsf{sk}_{\mathcal{G}_1}$ and $\mathsf{pk}_{\mathcal{G}_2}, \mathsf{sk}_{\mathcal{G}_2}$ under Π_{S}. Polling officers $(\mathcal{P}_j)_{j\in[\ell]}$ also generate public/private keys $(\mathsf{pk}_{\mathcal{P}_j}, \mathsf{sk}_{\mathcal{P}_j})_{j\in[\ell]}$ under Π_{S}. Mix-servers $(\mathcal{M}_k)_{k\in[m]}$ jointly run the $\Pi_{\mathsf{TM}}.\mathsf{Keygen}$ protocol of the traceable mixnet to generate the mixnet public key mpk and individual secret keys $(\mathsf{msk}^{(k)})_{k\in[m]}$ for each $(\mathcal{M}_k)_{k\in[m]}$. An official candidate list $(\mathsf{cand}_0, \ldots, \mathsf{cand}_{\alpha-1})$ is created such that cand_a denotes the a^{th} candidate.

4.2.2 Ballot Design

Our ballot (Figure 3 - left) customises the Scratch & Vote ballot [1] for dual voting and BMD support. It consists of two halves connected by a perforated line. The left half serves as the VVPR, while the right half serves as the voter receipt. These halves are unlinkable after the vote is cast.

The left half includes a randomly drawn ballot identifier bid from $\mathbb{Z}_q$. It displays a circular rotation of the official candidate list. For each $w \in \{0, \ldots, \alpha - 1\}$, row w corresponds to the candidate $\mathsf{cand}_{\mathsf{bid}+w \bmod \alpha}$. For example, if the official candidate list is ("Alice", "Bob", "Carol"), and $\mathsf{bid} = 302$, the candidate printed on row $w = 1$ would be $\mathsf{cand}_{302+1 \bmod 3} = \mathsf{cand}_0$ (i.e., "Alice"). The right half contains corresponding encryptions c_w obtained by running the $\Pi_{\mathsf{TM}}.\mathsf{Enc}$ algorithm on input $\bar{v}_w = \mathsf{bid} + w$, except that they do not include the ρ_{γ_w} component; this is added during the Tally protocol. We call values $\bar{v}_w \in \mathbb{Z}_q$ the *extended votes* and values $v_w := \bar{v}_w \bmod \alpha \in [\alpha]$ the *raw votes*. The randomnesses r_w used in creating encryption c_w are kept secret and placed under a detachable scratch surface on the ballot.

Both halves feature a designated gray area at the top. During the Cast protocol, the BMD prints the voter-selected w in this gray area on both halves. Additionally, the right half includes a polling booth identifier j for the designated polling booth of the ballot, while the left half contains its commitment $\gamma_{\mathsf{booth}} = g^j h^{r_{\mathsf{booth}}}$. Randomness $r_{\mathsf{booth}} \xleftarrow{\$} \mathbb{Z}_q$ is also put under a separate scratch surface. The commitment γ_{booth} is revealed when the polling booth of a disputed VVPR needs to be identified in the TallyAudit protocol.

Due to the size of encryptions c_w (around 20 KB each [2]), they may not fit within standard QR codes on the paper ballot. However, conceptually, the actual encryptions could be stored in a backend server, with only a binding hash printed on the ballot. For simplicity, we ignore this complication.

4.2.3 Ballot Generation

During the BallotGen protocol, a ballot is jointly generated by $\mathcal{G}_1$ and $\mathcal{G}_2$ to hide the voter-vote association from any one of them (see Figure 4). $\mathcal{G}_1$, who selects the ballot secrets, does not learn the encryptions printed on the receipt half and cannot match voters to their ballot secrets, while $\mathcal{G}_2$, who creates the receipt half, does not know the ballot secrets.

$\mathcal{G}_2$ knows the destination booth j but keeps it hidden from $\mathcal{G}_1$ to hide booth-level voting statistics. It generates a commitment γ_{booth} for j and shares it with $\mathcal{G}_1$ (lines 1-2), who prints it on the left half of the ballot. $\mathcal{G}_1$ generates a secret ballot identifier bid and signs it (lines 2-3), computes $\bar{v}_w = \mathsf{bid} + w$ for each w and accordingly prints candidate names on the left half and randomnesses r_w under a scratch surface on the right half (lines 6,10). It then sends the partially printed ballot to $\mathcal{G}_2$, keeping the left half hidden. This can be done, e.g., by folding the ballot along the perforation line, sealing it and letting $\mathcal{G}_2$ print its contents on the *back side* of the right half. It also sends encryptions of each $\bar{v}_w$ under $\Pi_{\mathsf{TM}}.\mathsf{Enc}$, except the ρ_{γ_w} components, to $\mathcal{G}_2$ (lines 7-9,12).

$\mathcal{G}_2$ re-randomises the obtained commitments/encryptions and homomorphically computes fresh shares of $\bar{v}_w$ and the commitment randomnesses using the additive homomorphism of E in $\mathbb{Z}_q$ (lines 13-19). It then prints these re-randomised encryptions on the right half of the received ballot and signs them. The re-randomisation ensures that $\mathcal{G}_1$ cannot identify the ballot corresponding to a voter from their receipt. The commitment randomness r_{booth} of γ_{booth} is printed on another scratch surface on the right half.

4.2.4 Vote Casting

The Cast protocol for voter $\mathcal{V}_i$ at booth j is as follows (Figure 3 - center):

- *Ballot pick-up and eligibility verification:* $\mathcal{V}_i$ picks up a random sealed ballot from a set of ballots kept at the polling booth. The polling officer $\mathcal{P}_j$ verifies $\mathcal{V}_i$'s eligibility in the presence of polling agents and allows $\mathcal{V}_i$ to proceed to a private room containing a BMD $\mathcal{D}_j$.

- *Vote casting:* $\mathcal{V}_i$ feeds the top gray region of the ballot to $\mathcal{D}_j$ and presses a button on the onscreen display to select w corresponding to her preferred candidate. We denote the voter's chosen w as $\dot{w}$. $\mathcal{D}_j$ can only access the top gray region for printing and cannot read any part of the ballot (it should not have any attached scanner or camera). $\mathcal{D}_j$ prints $\dot{w}$ on both the left and the right halves of this gray region.

  $\mathcal{V}_i$ needs to verify that indeed her intended choice is printed on both the halves. If satisfied, $\mathcal{V}_i$ separates the left half of the marked ballot (the VVPR), folds it and drops it into a physical ballot box $\mathcal{B}_j$ kept near $\mathcal{P}_j$ such that $\mathcal{P}_j$ can verify that the voter dropped an official VVPR. The right half (the receipt) is given to $\mathcal{P}_j$ for scanning. If

1 $\mathcal{G}_2$: $r_{\mathsf{booth}} \xleftarrow{\$} \mathbb{Z}_q$; $\gamma_{\mathsf{booth}} \leftarrow g_1^j h_1^{r_{\mathsf{booth}}}$
2 $\mathcal{G}_2$: send γ_{booth} to $\mathcal{G}_1$
3 $\mathcal{G}_1$: $\mathsf{bid} \xleftarrow{\$} \mathbb{Z}_q$
4 $\sigma_{\mathsf{bid}} \leftarrow \Pi_{\mathsf{S}}.\mathsf{Sign}(\mathsf{sk}_{\mathcal{G}_1}, \mathsf{bid})$
5 **for** $w \in \{0, \ldots, \alpha - 1\}$**:**
6 $\bar{v}_w \leftarrow \mathsf{bid} + w$; $r_w \xleftarrow{\$} \mathbb{Z}_q$; $\gamma_w \leftarrow g_1^{\bar{v}_w} h_1^{r_w}$
7 $\epsilon_{\bar{v}_w} \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{Pa}}, \bar{v}_w)$; $\epsilon_{r_w} \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{Pa}}, r_w)$ // interpret v_w, r_w as elements of $\mathbb{Z}_N$
8 $(\bar{v}_w^{(k)})_{k\in[m]} \leftarrow \mathsf{Share}_{(m,m)}(\bar{v}_w)$; $(r_w^{(k)})_{k\in[m]} \leftarrow \mathsf{Share}_{(m,m)}(r_w)$
9 $(\mathsf{ev}_w^{(k)})_{k\in[m]} \leftarrow (\mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, \bar{v}_{w_i}^{(k)}))_{k\in[m]}$; $(\mathsf{er}_{w_i}^{(k)})_{k\in[m]} \leftarrow (\mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, r_{w_i}^{(k)}))_{k\in[m]}$
10 print ballot's left half ($\mathsf{cand}_{\bar{v}_w \bmod \alpha}$ on row w) and $(r_w)_{w\in\{0,\ldots,\alpha-1\}}$ under a scratch surface as per Fig. 3 - left
11 send the ballot to $\mathcal{G}_2$ with its left half sealed
12 send $(\epsilon_{\bar{v}_w}, \gamma_w, (\mathsf{ev}_w^{(k)}, \mathsf{er}_w^{(k)})_{k\in[m]}, \epsilon_{r_w})_{w\in\{0,\ldots,\alpha-1\}}$ to $\mathcal{G}_2$ electronically
13 $\mathcal{G}_2$: **for** $w \in \{0, \ldots, \alpha - 1\}$**:**
14 $r'_w \xleftarrow{\$} \mathbb{Z}_q$; $\gamma'_w \leftarrow \gamma_w h_1^{r'_w}$
15 $\epsilon'_{\bar{v}_w} \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{REnc}(\mathsf{pk}_{\mathsf{Pa}}, \epsilon_{\bar{v}_w})$; $\epsilon'_{r_w} \leftarrow \mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{REnc}(\mathsf{pk}_{\mathsf{Pa}}, \epsilon_{r_w})$ // $\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{REnc}(\mathsf{pk}_{\mathsf{Pa}}, \epsilon) = \epsilon\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{Pa}}, 0)$
16 $(v'^{(k)}_w)_{k\in[m]} \leftarrow \mathsf{Share}_{(m,m)}(0)$; $(r'^{(k)}_w)_{k\in[m]} \leftarrow \mathsf{Share}_{(m,m)}(r'_w)$
17 $(\mathsf{ev}'^{(k)}_w)_{k\in[m]} \leftarrow (\mathsf{ev}_w^{(k)} \cdot \mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, v'^{(k)}_w))_{k\in[m]}$
18 $(\mathsf{er}'^{(k)}_w)_{k\in[m]} \leftarrow (\mathsf{er}_w^{(k)} \cdot \mathsf{E}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{M}_k}, r'^{(k)}_w))_{k\in[m]}$
19 $c_w := (\epsilon'_{\bar{v}_w}, \gamma'_w, (\mathsf{ev}'^{(k)}_w, \mathsf{er}'^{(k)}_w)_{k\in[m]}, \epsilon'_{r_w})$
20 $\sigma_c \leftarrow \Pi_{\mathsf{S}}.\mathsf{Sign}(\mathsf{sk}_{\mathcal{G}_2}, H((c_w)_{w\in\{0,\ldots,\alpha-1\}}))$, where H is a hash function
21 print ballot's right half and r_{booth} under another scratch surface as per Fig. 3 - left
22 store (j, r_{booth}) indexed by γ_{booth}

Figure 4: The BallotGen protocol for generating a ballot for booth j known only to $\mathcal{G}_2$.

not satisfied, $\mathcal{V}_i$ shreds the marked ballot and raises a dispute. In this case, $\mathcal{V}_i$ is allowed to re-vote. Note that the vote casting phase can also completely avoid the BMD and require the voter to hand-mark the two ballot halves, but this design is prone to more voter errors (see Section 2).

- *Receipt scanning:* $\mathcal{P}_j$ checks that the scratch surface on $\mathcal{V}_i$'s receipt is intact, i.e., the ballot secrets are not compromised, and shreds the scratch region in front of $\mathcal{V}_i$. From the scanned receipt, $\mathcal{P}_j$ extracts $c_{\dot{w}}$ and uploads $(\mathsf{vid}_i, j, c_{\dot{w}})$ to C, along with $\sigma_{\mathsf{vid}_i} \leftarrow \Pi_{\mathsf{S}}.\mathsf{Sign}(\mathsf{sk}_{\mathcal{P}_j}, (\mathsf{vid}_i, j, c_{\dot{w}}))$. $\mathcal{P}_j$ also affixes vid_i to $\mathcal{V}_i$'s receipt, stamps it for authenticity, and returns it to $\mathcal{V}_i$.

Chain voting and randomisation attacks. With minor modifications, these sophisticated coercion attacks can also be handled. For chain voting, $\mathcal{P}_j$ can stamp a serial number on the receipt half of the sealed ballot after identity verification to prevent the use of rogue ballots. This number is matched before accepting the voter's receipt. Under a *randomisation attack*, voters may be asked to choose a fixed $\dot{w}$, thereby randomising their votes. To counter this, voters should be allowed to choose their ballots in a private room. The ballot cover should contain a detachable slip showing the candidate order, allowing coerced voters to choose a ballot so that they can vote for their preferred candidate while producing the $\dot{w}$ satisfying the coercer. Before proceeding to $\mathcal{P}_j$, the voter should detach the slip.

4.2.5 Vote Tallying

Post polling, $(\mathcal{M}_k)_{k\in[m]}$ process the tuples $\{(\mathsf{vid}_i, j_i, c_i)\}_{i=0}^{n-1}$ uploaded on C by $(\mathcal{P}_j)_{j\in[\ell]}$, where c_i denotes $c_{\dot{w}}$ for the i^{th} voter (Figure 3 - right). $(\mathcal{M}_k)_{k\in[m]}$ proceed as per Figure 5 where they first add the ρ_{γ_i} components to the encryptions c_i by engaging in a distributed NIZK proof of knowledge (lines 2-9) and then processing $(c_i)_{i\in[n]}$ through the traceable mixnet's Mix protocol (line 13). At the end, the permuted extended votes $(\bar{v}'_i)_{i\in[n]}$ are obtained from which the raw votes are computed (line 14). Both extended and raw votes are published on V.

The VVPRs from each polling booth's ballot box are collected and mixed in a central facility. VVPRs are revealed to the public only after this mixing phase and post audit, to avoid leaking polling booth-level voting statistics. A VVPR containing ballot identifier bid and voter choice $\dot{w}$ can be matched with the corresponding decrypted vote by computing $\mathsf{bid} + \dot{w}$, finding it on V and checking if the corresponding raw vote matches the candidate name printed on the $\dot{w}^{\text{th}}$ row on the VVPR.

1	$(\mathcal{M}_k)_{k\in[m]}$:	**for** $i \in [n]$**:**
2		$\bar{v}_i^{(k)} \leftarrow \mathsf{E.Dec}(\mathsf{sk}_{\mathcal{M}_k}, \mathsf{ev}_i^{(k)})$ // decryption under E
3		$r_i^{(k)} \leftarrow \mathsf{E.Dec}(\mathsf{sk}_{\mathcal{M}_k}, \mathsf{ev}_i^{(k)})$
4		// Generate a distributed NIZK PoK ρ_{γ_i} of the opening of γ_i using shares $(\bar{v}_i^{(k)}, r_i^{(k)})_{k\in[m]}$
5		$r_{v_i}^{(k)}, r_{r_i}^{(k)} \xleftarrow{\$} \mathbb{Z}_q$; $a_i^{(k)} \leftarrow g_1^{r_{v_i}^{(k)}} h_1^{r_{r_i}^{(k)}}$; publish $a_i^{(k)}$.
6		$\mathsf{c}_i \leftarrow H(\gamma_i \Vert \prod_{k\in[m]} a_i^{(k)})$; $z_{\bar{v}_i}^{(k)} \leftarrow r_{v_i}^{(k)} - \bar{v}_i^{(k)}\mathsf{c}_i$; $z_{r_i}^{(k)} \leftarrow r_{r_i}^{(k)} - r_i^{(k)}\mathsf{c}_i$; publish $z_{\bar{v}_i}^{(k)}, z_{r_i}^{(k)}$.
7		$\rho_{\gamma_i} := (a_i, \mathsf{c}_i, (z_{\bar{v}_i}, z_{r_i})) \leftarrow (\prod_{k\in[m]} a_i^{(k)}, H(\gamma_i \Vert \prod_{k\in[m]} a_i^{(k)}), (\sum_{k\in[m]} z_{\bar{v}_i}^{(k)}, \sum_{k\in[m]} z_{r_i}^{(k)}))$.
8		// ρ_{γ_i} can be verified by checking if $\mathsf{c}_i \stackrel{?}{=} H(\gamma_i \Vert a_i)$ and $\gamma_i^{\mathsf{c}_i} g_1^{z_{\bar{v}_i}} h_1^{z_{r_i}} \stackrel{?}{=} a_i$.
9		update c_i by inserting ρ_{γ_i} into it
10		**endfor**
11		// Mixing protocol to generate permuted *extended* votes
12		// Each $\mathcal{M}_k$ gets secret input $\mathsf{msk}^{(k)}$ and secret output $\omega^{(k)}$ (see Section **??**)
13	$(\mathcal{M}_k)_{k\in[m]}$:	$(\bar{v}'_i)_{i\in[n]}, (\mathcal{M}_k[\![\omega^{(k)}]\!])_{k\in[m]} \leftarrow \Pi_{\mathsf{TM}}.\mathsf{Mix}(\mathsf{mpk}, (c_i)_{i\in[n]}, (\mathcal{M}_k[\![\mathsf{msk}^{(k)}]\!])_{k\in[m]})$
14		$(v'_i)_{i\in[n]} \leftarrow (\bar{v}'_i \bmod \alpha)_{j\in[n]}$
15		publish $(\bar{v}'_i)_{i\in[n]}, (v'_i)_{i\in[n]}$ to V; $\mathcal{M}_k$ stores $\omega^{(k)}$

Figure 5: The Tally protocol involving $(\mathcal{M}_k)_{k\in[m]}$ on input $\mathsf{mpk}, (c_i)_{i\in[n]}$ and $\mathcal{M}_k$'s input $\mathsf{msk}^{(k)}$ containing $\mathsf{sk}_{\mathcal{M}_k}$.

4.2.6 Ballot and Receipt Audits

In the BallotAudit protocol, a statistically significant number of ballots at each polling booth must be audited to keep the probability ϵ_{b} of a cast-as-intended failure (see Section 3) small. Ballot audits can happen before, during or after polling, and even be initiated by voters. When auditing a ballot, its sealed cover is opened and secrets under its scratch surfaces are revealed. For each $w = 0 \ldots \alpha - 1$, it is checked that encryption c_w is created correctly on message $\mathsf{bid} + w$ using r_w and the candidate name printed at row w is $\mathsf{cand}_{\mathsf{bid}+w \bmod \alpha}$, where bid is looked up from the left half and r_w from the scratch surface. Further, it is checked that $\gamma_{\mathsf{booth}} \stackrel{?}{=} g_1^j h_1^{r_{\mathsf{booth}}}$, where j is the audited booth's identifier and r_{booth} is obtained from the scratch surface, and that signatures by $\mathcal{G}_1, \mathcal{G}_2$ verify. Since the secrets of audited ballots are revealed, audited ballots cannot be used for vote casting and must be spoiled.

Similarly, in the ReceiptAudit protocol, a statistically significant number of voter receipts from each polling booth must be checked for their existence on list C to keep ϵ_{r} small. All audited receipts should be uploaded to R to aid audit and recovery.

4.2.7 Tally Audit

Our tally audit protocol (see Figure 6) depends on BTraceIn and BTraceOut queries of a traceable mixnet (see Section **??**). Given (R, C, V, P), first, all input-phase failures are marked (lines 1-3). Here, as per the discussion in Section 2, we mark all receipts/encrypted votes from a booth as failed if any one of them fails and the encrypted votes as failed if the ballot audit at that booth failed. For marking mixing phase failures on C and V, we run the BTraceIn/BTraceOut queries against the complete set of entries on V and C respectively (lines 4-5). Output-phase failures are marked by comparing the VVPRs with the decrypted extended votes (lines 6-7).

If the total number of failures is less than the winning margin, then $J^* = \emptyset$ and $V^* = V$ are reported, signalling that no rerun is required (lines 9-10). Otherwise, polling booths contributing all the failing items are identified: for receipts and encrypted votes, the booth identifiers directly exist on R and C (line 12); for VVPRs without an electronic entry, they are identified by asking $\mathcal{G}_2$ to open the opening of γ_{booth} printed on the VVPR (line 13); for decrypted votes, a BTraceOut query against the set of ciphertexts cast at a booth j is run for all booths $j \in [\ell]$ (lines 14-18). The set of all such booths is reported in J^* (line 19). The decrypted votes V^* contributed by the good booths are obtained by running another BTraceOut query against the entries on C contributed by booths outside of J^* (line 21).

4.2.8 Recovery

The suggested recovery is to rerun the election only on booths in J^* and later merge this tally with the tally reported in V^*. However, if J^* is small, one can also consider rerunning on a few randomly selected good booths too, to avoid specialised targeting of the booths in J^*. Further, the general approach of TraceIn/TraceOut queries can also support other recovery options for dual voting systems. For example, one can immmediately recover from case

1 $J_{\mathrm{FC}} \leftarrow \{j \in [\ell] \mid \mathsf{BallotAudit}$ fails at booth $j\}$
2 $R_{\mathrm{FI}} \leftarrow \{r \in R \mid r$ fails against C under $\mathsf{FI}_1, \mathsf{FI}_2\}$; $R_{\mathrm{FI}} \leftarrow \{(\mathsf{vid}, j, c) \in R \mid (\mathsf{vid}', j, c') \in R_{\mathrm{FI}}\}$
3 $C_{\mathrm{FI}} \leftarrow \{c \in C \mid c$ fails against R under $\mathsf{FI}_2\}$; $C_{\mathrm{FI}} \leftarrow \{(\mathsf{vid}, j, c) \in C \mid (\mathsf{vid}', j, c') \in C_{\mathrm{FI}} \vee j \in J_{\mathrm{FC}}\}$
4 $C_{\mathrm{FM}} \leftarrow \{c_i\}_{i\in[n_c]} \setminus \mathsf{BTraceIn}(\mathsf{mpk}, (c_i)_{i\in[n_c]}, (\bar{v}'_i)_{i\in[n_v]}, [n_c], [n_v], (\mathcal{M}_k[\![\mathsf{msk}^{(k)}, \omega^{(k)}]\!])_{k\in[m]}, \mathcal{A}[\![]\!])$
5 $V_{\mathrm{FM}} \leftarrow \{v'_i\}_{i\in[n_v]} \setminus \mathsf{BTraceOut}(\mathsf{mpk}, (c_i)_{i\in[n_c]}, (\bar{v}'_i)_{i\in[n_v]}, [n_c], [n_v], (\mathcal{M}_k[\![\mathsf{msk}^{(k)}, \omega^{(k)}]\!])_{k\in[m]}, \mathcal{A}[\![]\!])$
6 $V_{\mathrm{FO}} \leftarrow \{\bar{v} \in V \mid \bar{v}$ fails against P under $\mathsf{FO}_1, \mathsf{FO}_3, \mathsf{FO}_4\}$
7 $P_{\mathrm{FO}} \leftarrow \{p \in P \mid p$ fails against V under $\mathsf{FO}_2, \mathsf{FO}_3, \mathsf{FO}_5\}$
8 $R_{\mathrm{F}} \leftarrow R_{\mathrm{FI}}$; $C_{\mathrm{F}} \leftarrow C_{\mathrm{FI}} \cup C_{\mathrm{FM}}$; $V_{\mathrm{F}} \leftarrow V_{\mathrm{FM}} \cup V_{\mathrm{FO}}$; $P_{\mathrm{F}} \leftarrow P_{\mathrm{FO}}$
9 **if** $|R_{\mathrm{F}}| + |C_{\mathrm{F}}| + |V_{\mathrm{F}}| + |P_{\mathrm{F}}| <$ winning margin calculated from V:
10 $\quad J^* \leftarrow \emptyset$; $V^* \leftarrow V$
11 **else:**
12 $\quad \mathsf{badbooths}_r \leftarrow \{j \mid (\mathsf{vid}, j, c) \in R_{\mathrm{F}}\}$; $\mathsf{badbooths}_c \leftarrow \{j \mid (\mathsf{vid}, j, c) \in C_{\mathrm{F}}\}$
13 $\quad \mathsf{badbooths}_p \leftarrow \{j \mid \mathcal{G}_2$ supplies (j, r_{booth}) to $\mathcal{A}$ for γ_{booth} printed on some $p \in P_{\mathrm{F}}$ s.t. $\gamma_{\mathsf{booth}} = g_1^j h_1^{r_{\mathsf{booth}}}\}$
14 $\quad \mathsf{badbooths}_v \leftarrow \emptyset$
15 $\quad$ **for** j **in** $[\ell]$**:**
16 $\quad\quad$ // I_j denotes indices of booth j's entries in C; $I'_{V_{\mathrm{F}}}$ denotes indices of V_{F} entries on V
17 $\quad\quad V_{\mathrm{F}_j} \leftarrow \mathsf{BTraceOut}(\mathsf{mpk}, (c_i)_{i\in[n_c]}, (v'_i)_{i\in[n_v]}, I_j, I'_{V_{\mathrm{F}}}, (\mathcal{M}_k[\![\mathsf{msk}^{(k)}, \omega^{(k)}]\!])_{k\in[m]}, \mathcal{A}[\![]\!])$
18 $\quad\quad$ **if** $V_{\mathrm{F}_j} \neq \emptyset$**:** $\mathsf{badbooths}_v \leftarrow \mathsf{badbooths}_v \cup \{j\}$
19 $\quad J^* \leftarrow \mathsf{badbooths}_r \cup \mathsf{badbooths}_c \cup \mathsf{badbooths}_p \cup \mathsf{badbooths}_v$
20 $\quad$ // $I_{\mathsf{goodbooths}}$ denotes indices of booths outside J^* in C; $I'_{V\setminus V_{\mathrm{F}}}$ denotes indices of entries outside V_{F} on V
21 $\quad V^* \leftarrow \mathsf{BTraceOut}(\mathsf{mpk}, (c_i)_{i\in[n_c]}, (v'_i)_{i\in[n_v]}, I_{\mathsf{goodbooths}}, I'_{V\setminus V_{\mathrm{F}}}, (\mathcal{M}_k[\![\mathsf{msk}^{(k)}, \omega^{(k)}]\!])_{k\in[m]}, \mathcal{A}[\![]\!])$
22 **return** J^*, V^*

Figure 6: The TallyAudit protocol involving $\mathcal{A}$, $(\mathcal{M}_k)_{k\in[m]}$ and $\mathcal{G}_2$ with public input (R, C, V, P), each $\mathcal{M}_k$'s input its mixnet secret key $\mathsf{msk}^{(k)}$ and witness $\omega^{(k)}$ output by the traceable mixnet during the Tally protocol, and $\mathcal{G}_2$'s input being (j, r_{booth}) stored indexed by γ_{booth} at the end of the BallotGen protocol.

FO_3 if a TraceOut query is run for the decrypted vote against the set of encrypted votes that successfully matched with voter receipts. If the answer is yes, then it provides solid evidence that the VVPR is wrong, without leaking any additional information. A similar query run for the VVPR provides solid evidence that the electronic vote was wrong. Of course, what queries to allow must be carefully decided depending on the situation to best optimise the recoverability-secrecy tradeoff.

5 Security Analysis

Theorem 1. *Under the DL assumption in $\mathbb{G}_1$, the n-SDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ [7] and the EUF-CMA security of Π_{S}, the* OpenVoting *protocol is recoverable as per Definition 1.*

Proof (Sketch). We focus on the event that for each booth, BallotAudit passing implies that all receipts correctly captured voter intents and ReceiptAudit passing implies that all receipts were correctly uploaded. This event happens with probability $1 - \ell(\epsilon_{\mathsf{b}} + \epsilon_{\mathsf{r}})$.

Let J^*, V^* be $\mathcal{A}$'s output in the TallyAudit protocol. From Figure 6, we consider the two cases: first when the branch on line 8 is taken and the second when it is not taken. In the first case, $J^* = \emptyset$ and thus we must show that the number of failures in (R^*, C^*, V^*, P^*) is less than the winning margin, where $R^* = R$, $C^* = C$ and $P^* = P$ for $J^* = \emptyset$ and $V^* = V$ by line 10. By the condition on line 9, the number of *reported* failures is less than the winning margin. By the soundness of Π_{TM} under the stated assumptions [2], sets C_{FM} and V_{FM} correctly represent the set of true failures. This, combined with the definitions of R_{FI}, C_{FI}, V_{FO} and P_{FO}, implies that the number of real failures in (R^*, C^*, V^*, P^*) is less than the winning margin.

In the second case, J^* is, as required, exactly the non-empty set of booths that contributed some failing item in R_{F}, C_{F} (by the definitions on line 12), P_{F} (by line 13 and the computational binding of Pedersen commitments under the DL assumption in $\mathbb{G}_1$) or V_{FO} (by lines 14-18 and the soundness property of Π_{TM}; note that V_{FM} entries in V_{F} are mix-server errors and, as required, are not reported here). Finally, by line 21 and the soundness of Π_{TM}, V^* is exactly the set of votes decrypted from encrypted votes sent by booths outside J^*. Thus, by the definitions of R^*, C^* and P^*, (R^*, C^*, V^*, P^*) does not contain any failures. □

Theorem 2. *Under the DDH assumption in $\mathbb{G}_1$ and the DCR assumption [19], the* OpenVoting *protocol satisfies vote secrecy with recoverability as per Definition 2 in the random oracle model.*

Proof (Sketch). If the adversary corrupts $\mathcal{G}_2$ but not $\mathcal{G}_1$, then it does not learn the ballot secrets of ballots used by $\mathcal{V}_{i_0}$ and $\mathcal{V}_{i_1}$ by the perfect hiding of Pedersen commitments under the DDH assumption and the IND-CPA security of Paillier schemes E and $\mathsf{E}^{\mathsf{th}}_{\mathsf{Pa}}$ under the DCR assumption (see Figure 4). Post-printing, ballots get sealed and are opened only by the voter during vote casting, where the adversary-controlled BMD does not see any information about the ballot used. The receipts and the tallying protocol does not reveal any information to the corrupted mix-servers by the secrecy property of Π_{TM} under the stated assumptions [2]. VVPRs are collected after mixing and the ballot identifiers used therein cannot be linked to the identifiers of $\mathcal{V}_{i_0}$ and $\mathcal{V}_{i_1}$. Finally, during the $\mathsf{TallyAudit}$ protocol, it is required that if the adversary causes a failure in either the receipt, encrypted vote or VVPR contributed by $\mathcal{V}_{i_0}$'s booth j_0 then it should also cause a failure in $\mathcal{V}_{i_1}$'s booth j_1. Thus, sets R_{FI} to P_{FO} in Figure 6 do not help it distinguish between the two worlds. Outputs V_{F_j} do not help because for each failure in booth j_0, the adversary is required to create an identical failure in booth j_1. Further, the partial tally V^* includes either both v_0, v_1 or none of them. The secrecy property of Π_{TM} ensures that no additional information beyond the query outputs is revealed.

If the adversary corrupts $\mathcal{G}_1$ but not $\mathcal{G}_2$, then it obtains ballot secrets but it cannot identify which of $\mathcal{V}_{i_0}$ or $\mathcal{V}_{i_1}$ used which ballot. The rest of the proof is similar. □

6 Conclusion and Future Work

We have introduced and formalised the notion of recoverability and secrecy for dual voting protocols and suggested a protocol that achieves this notion. Based on existing reports for the underlying traceable mixnet construction, the total time taken by the recovery process remains within a few hours for $n = 10000$ ciphertexts, which can be optimised further using the construction's high degree of task parallelism [2].

Although we have shown our protocol's recoverability properties, the potential non-termination of the revoting process during vote casting seems like an inherent limitation of BMD protocols and designing voting frontends that overcome this limitation yet remain usable and minimise voter errors appears to be a challenging open problem. Further, although we have focused on recoverability for first-past-the-post voting where exact winning margins are computable, extending to other more complex voting rules also appears to be an interesting avenue for future work.

Acknowledgments. Prashant Agrawal is supported by the Pankaj Jalote Doctoral Grant. Abhinav Nakarmi was supported by a research grant from MPhasis F1 Foundation.

References

[1] Ben Adida and Ronald L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. In *WPES*, pages 29–40, 2006.

[2] Prashant Agrawal, Abhinav Nakarmi, Mahavir Prasad Jhawar, Subodh Sharma, and Subhashis Banerjee. Traceable mixnets, 2023.

[3] Mukulika Banerjee. Blocking the introduction of the Totaliser is not good for the secret ballot in India. LSE South Asia Center Blog, 2017. [Online; accessed 19-May-2023].

[4] Susan Bell, Josh Benaloh, Michael D. Byrne, Dana Debeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-vote: A Secure, Transparent, Auditable, and Reliable voting system. In *EVT/WOTE*, 2013.

[5] Josh Benaloh. Administrative and public verifiability: can we have both? In *EVT*, pages 5:1–5:10, 2008.

[6] Matthew Bernhard, Josh Benaloh, J. Alex Halderman, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, Poorvi L. Vora, and Dan S. Wallach. Public evidence from secret ballots. In *EVOTE-ID*, pages 84–109, 2017.

[7] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.

[8] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: end-to-end voter-verifiable optical-scan voting. *IEEE S&P*, 6(3):40–46, 2008.

[9] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *Intl. Jr. Inf. Security*, 9:371–385, 2010.

[10] Yvo Desmedt. Threshold cryptography. *European Tr. Telecommunications*, 5(4):449–458, 1994.

[11] Aleks Essex and Jeremy Clark. Punchscan in practice: an E2E election case study. In *WOTE*, 2007.

[12] Aleksander Essex, Christian Henrich, and Urs Hengartner. Single layer optical-scan voting with fully distributed trust. In *EVOTE-ID*, pages 122–139, 2012.

[13] Niko Farhi. An implementation of dual (paper and cryptograhic) voting system. Master's thesis, The Blatavnik School of Computer Sciences, Tel Aviv University, 2013.

[14] Thomas Haines and Johannes Müller. SoK: techniques for verifiable mix nets. In *CSF*, pages 49–64, 2020.

[15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. pages 526–535, 2010.

[16] Mark Lindeman, Philip B. Stark, and Vincent S. Yates. BRAVO: Ballot-polling Risk-limiting Audits to Verify Outcomes. In *EVT/WOTE*, 2012.

[17] David Lundin and Peter Y. A. Ryan. Human readable paper verification of Prêt à Voter. In *ESORICS*, pages 379–395, 2008.

[18] NDI. The constitutionality of electronic voting in Germany, 2019. [Accessed June 8, 2019].

[19] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Intl. Conf. Theory and Applications of Cryptographic Techniques*, pages 223–238, 1999.

[20] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.

[21] Philip B. Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *EVT/WOTE*, 2010.

[22] Stefan Popoveniuc and Andrew Regenscheid. Sigma ballots. In *EVOTE-ID*, volume P-167, pages 179–190, 2010.

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

[24] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. PrêT à Voter: a voter-verifiable voting system. *Tr. Information Forensics and Security*, 4(4):662–673, 2009.

[25] Philip B. Stark. Conservative statistical post-election audits. In *The Annals of Applied Statistics*, volume 2, pages 550–581, 2008.