Camel: E2E Verifiable Instant Runoff Voting without Tallying Authorities

Luke Harrison
University of Warwick
United Kingdom
l.harrison.3@warwick.ac.uk

Samiran Bag
University of Warwick
United Kingdom
Samiran.Bag@warwick.ac.uk

Feng Hao
University of Warwick
United Kingdom
feng.hao@warwick.ac.uk

ABSTRACT

Instant Runoff Voting (IRV) is one example of ranked-choice voting. It provides many known benefits when used in elections, such as minimising vote splitting, ensuring few votes are wasted, and providing resistance to strategic voting. However, the voting and tallying procedures for IRV are much more complicated than those of plurality and are both error-prone and tedious. Many automated systems have been proposed to simplify these procedures in IRV. Some of these also employ cryptographic techniques to protect the secrecy of ballots and enable verification of the tally. Nearly all of these cryptographic systems require a set of trustworthy tallying authorities (TAs) to perform the decryption of votes and/or running of mix servers, which adds significant complexity to the implementation and election management. We address this issue by proposing Camel: an E2E verifiable solution for IRV that requires no TAs. Camel employs a novel representation and a universally verifiable shifting procedure for ballots that facilitate the elimination of candidates as required in an IRV election. We combine these with a homomorphic encryption scheme and zero-knowledge proofs to protect the secrecy of the ballots and enable any party to verify the well-formedness of the ballots and the correctness of the tally in an IRV election. We examine the security of Camel and prove it maintains ballot secrecy by limiting the learned information (namely the tally) against a set of colluding voters.

KEYWORDS

E2E verifiability; Self-enforcing e-voting; Instant Runoff Voting.

1 INTRODUCTION

Instant Runoff Voting (IRV), commonly referred to as the Alternative Vote (AV) in the United Kingdom, is an example of ranked-choice voting. In these methods, a voter must rank a list of candidates in order of their preference. Once all votes have been cast in an IRV election, the most-preferred choice on each vote is examined, and the candidate that receives the fewest most-preferred placements amongst all votes is eliminated from the election and removed from the votes [1]. This procedure repeats until a candidate receives 50% of all most-preferred placements, at which point they are declared the winner. IRV has been used for many local elections in the United States, including in San Francisco, Oakland, Santa Fe and New York City, as well as for statewide elections in Maine and Alaska [2]. It has also seen use in the national legislature of Australia, the presidential elections in Ireland [3], and the party leadership elections in the United Kingdom [4].

Many benefits have been listed for adopting IRV in elections compared to other voting systems. IRV provides more representative outcomes and helps to avoid vote splitting, where two candidates with similar ideologies may split the vote of their base between themselves (this problem has been noted particularly for plurality voting [5]). IRV has also been shown to commonly elect the Condorcet winner: the candidate that is the most socially-optimal [6]. Unlike Condorcet voting however, candidates in IRV must have sufficiently broad appeal and enough most-preferred placements to progress in an election without elimination. Strategic voting is also less of a concern when using IRV; although the complete avoidance of strategic voting in IRV is impossible (due to the Gibbard-Satterthwaite theorem [7, 8]), it is noted that IRV is highly resistant to strategic voting when compared against other voting methods including plurality, Borda count, approval voting and range voting, with only a small difference in comparison to the Woodall, Benham, Smith-AV and Tideman methods [9].

Despite all these benefits, it can be difficult to adopt IRV for practical elections. This is largely due to the vote counting and elimination procedures, which can be complex and error-prone to perform by hand. We may instead simplify and automate these processes by adopting an e-voting solution for IRV through either voting machines or online voting. The UK Labour Party 2020 Leadership election used IRV with online ballots, although those without a verified email address were permitted to vote by postal ballot [4]. However, these systems for IRV are not verifiable; voters must assume that their vote has been tallied and recorded correctly, with no means to truly verify this for themselves. A malfunctioning or compromised voting server may alter a vote in a way that is imperceptible to the voter. We address this in our paper by proposing a fully verifiable system for IRV. Our solution builds upon the important notion of End-to-End (E2E) verifiability, which covers verifiability in three aspects: *cast as intended*, *recorded as cast*, and *tallied as recorded* [10].

E2E verifiable systems have been studied previously for IRV [11–15]. All of these systems rely upon a set of trustworthy parties known as *tallying authorities* (or TAs) to perform the cryptographic procedures necessary for computing the tallies, however, choosing and managing such TAs has proven to be a particularly difficult issue in practice [16]. To overcome this, we follow the notion of self-enforcing e-voting (SEEV) initiated by DRE-i [17] and DRE-ip [18]. SEEV aims to achieve E2E verifiability without the need for any TAs. To realise this, we adopt a similar approach to the cancellation of random factors used in DRE-ip in our design. DRE-ip only supports plurality voting, whilst our protocol supports IRV: a much more complicated electoral scheme than that of plurality. The major challenges in our work lie in proving that the IRV ballots have been constructed correctly and that the IRV voting procedure has been followed accordingly, with the correct tally computed, and appropriate candidates eliminated, from ballots at each stage of the tallying process. These challenges must be solved in a secure

manner that allows public verification of the ballots and the tally. We will explain how we have overcome these challenges through a new matrix representation of IRV ballots. This representation permits straightforward tallying of IRV ballots as well as verification of eliminated candidates through a novel shifting procedure.

In this paper, we propose a new voting protocol called "Camel"[1]. Camel is the first E2E verifiable e-voting system for IRV without tallying authorities. First, we explain the standard IRV procedure for paper ballots (§2) and then introduce a matrix representation of IRV ballots, together with a novel procedure to tally, eliminate and transfer votes in their electronic form. Based on the new matrix representation, we construct the Camel protocol, using zero-knowledge proofs to prove that all encrypted ballots are well-formed and that the tallying process is done according to the specification with public verifiability (§3). We present formal security proofs to prove the security of Camel (§4). Next, we compare Camel with related E2E verifiable solutions in the literature to demonstrate its feasibility for IRV elections in practice (§5). Finally, §6 concludes the paper.

2 PRELIMINARIES

2.1 Instant Runoff Voting

IRV requires each voter to rank n candidates for an election in order of preference. For instance, in an election between candidates A, B, C and D, a voter may choose to write down their vote as $[A : 1^{st} ; B : 3^{rd} ; C : 2^{nd} ; D : 4^{th}]$, meaning they prefer A to all other candidates, C to B and D (but not A), B to D (but not A or C) and D to no others. Given a collection of votes, the election authority then tallies all most-preferred choices amongst all votes to produce a collective ranking. For an election consisting of 60 voters, we might have the tally as $[A : 8 ; B : 17 ; C : 16 ; D : 19]$, meaning A received 8 most-preferred placements, B received 17, C received 16 and finally D received 19.

The tally of most-preferred placements is then examined to determine the candidate (or candidates) with the fewest placements. In the previous example, A would be this candidate. Once this candidate has been determined, it is then eliminated from all votes. In our example, any vote that placed A as their most-preferred choice would have their vote transferred to their next-most-preferred choice; for instance, the vote $[A : 1^{st} ; B : 3^{rd} ; C : 2^{nd} ; D : 4^{th}]$ would become $[B : 2^{nd} ; C : 1^{st} ; D : 3^{rd}]$. C becomes the new most-preferred choice. The votes are then re-tallied and the elimination procedure repeats over more rounds until one candidate is listed as the most-preferred choice on more than 50% of all votes and wins the election.

2.2 Permutation Matrices and Eliminations

IRV requires multiple passes over votes in order to eliminate candidates and determine a winner. Accordingly, we utilise permutation matrices for IRV ballots and eliminate candidates over multiple rounds, with each elimination being universally verifiable. We begin with an explanation of our method over plaintext votes, before presenting an E2E verifiable scheme in §3.

[1]Camels have multiple chambers in their stomach, which enables the processing of food in stages with the maximum extraction of nutrients and the minimum loss of water. This resembles how our protocol processes the tallying of votes in stages.

2.2.1 Setup. We first provide a setup procedure for our method. Let C denote a set of n candidates for an election. We assume each candidate in C is mapped to a unique value from $[1, n]$, e.g. $A \mapsto 1$, $B \mapsto 2, C \mapsto 3$ and $D \mapsto 4$. Let K denote a set of voters. Our scheme operates over $n-1$ tallying rounds (as we will explain). Initialise tally vectors $\mathbf{t}^{(m)} \leftarrow \left(t_1^{(m)}, \ldots, t_n^{(m)}\right)$ with $t_i^{(m)} = 0$ $(i \in [1, n])$, one for each round $m \in [1, n-1]$.

2.2.2 Vote Casting Phase. Our method operates over two phases. The first phase allows votes to be cast. Let the vector $\mathbf{a}_k = (a_1, \ldots, a_n)$ represent an n-ary permutation of the candidates for a voter $k \in K$, where a_i is preferred to a_j if $i < j$. We require voters to submit the full permutation $\mathbf{a}_k$ during this phase and not any partial or incomplete rankings. To facilitate tallying of votes in the subsequent phase, we transform $\mathbf{a}_k$ into a *permutation matrix* $\mathbf{V}_k = (v_{ij})_k$ where each row i in the matrix encodes the candidate a_i from $\mathbf{a}_k$. Suppose $\mathbf{a}_k$ is $(4, 2, 1, 3)$. We produce the following matrix from $\mathbf{a}_k$.

$$\begin{array}{c} 4 \\ 2 \\ 1 \\ 3 \end{array} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

Once all votes have been submitted and transformed into permutation matrices, we proceed onto the next phase.

2.2.3 Tallying Phase. The second phase of our method handles all tallying and elimination procedures within IRV and operates over $n-1$ rounds. We use $\mathbf{V}_k^{(m)}$ to denote the voting matrix $\mathbf{V}_k$ at round m. We initialise matrices $\mathbf{V}_k^{(1)}$ at the first round for all voters as their corresponding permutation matrices $\mathbf{V}_k$ from the previous Vote Casting phase.

In each round, we first check to see if any candidates must be eliminated. For the first round, there will be no such candidates. We compute the tally for the first round simply as $\mathbf{t}^{(1)} \leftarrow \sum_k (\mathbf{v}_1)_k^{(1)}$. Here, we use $(\mathbf{v}_i)_k^{(m)}$ to denote the row i from $\mathbf{V}_k^{(m)}$. Hence, $\mathbf{t}^{(1)}$ is computed as the sum over the first row of $\mathbf{V}_k^{(1)}$ for all voters, encapsulating all most-preferred choices. We then directly inspect $\mathbf{t}^{(1)}$ to determine if one candidate received more than 50% of preferences; if this is the case, we then terminate and announce a winner. Otherwise, we must mark a candidate for elimination that received the fewest preferences. Denote this candidate as $\alpha^{(1)}$ for the first round. We then progress onto the second round.

In the second round, we must eliminate the candidate $\alpha^{(1)}$ marked in the previous round from all the votes as required by IRV. To perform the eliminations, we introduce a new *shifting* procedure for permutation matrices. We inspect each matrix $\mathbf{V}_k^{(1)}$ from the previous round and locate the row in each that encodes $\alpha^{(1)}$. Let i denote the index of this row for any of the matrices. We delete this row i and *shift* all subsequent rows $i' > i$ up by one index. The result is a new matrix containing no eliminated candidates. We then compute the tally $\mathbf{t}^{(2)}$ for the second round as $\mathbf{t}^{(2)} \leftarrow \sum_k (\mathbf{v}_1)_k^{(2)}$ and mark a new candidate $\alpha^{(2)}$ for elimination in the following round. This procedure repeats until one candidate receives 50% of the votes in one of the rounds (in the worst case, we reach round

$n-1$ where only two candidates will remain). We illustrate this procedure over multiple rounds in Figure 1.

We give the full shifting procedure as Algorithm 1. This algorithm constructs a matrix $\mathbf{V}_k^{(m)}$ when given $\mathbf{V}_k^{(m-1)}$ from the previous round and the candidate $\alpha^{(m-1)}$ marked for elimination. In §3.3, we will show a zero-knowledge proof technique to ensure the shifting procedure is followed honestly without revealing votes. Once the shifting procedure completes and creates the voting matrix $\mathbf{V}_k^{(m)}$, we compute the tally at round m as $\mathbf{t}^{(m)} \leftarrow \sum_k (\mathbf{v}_1)_k^{(m)}$ and mark $\alpha^{(m)}$ for elimination at the start of the next round.

Algorithm 1 Shifting procedure for matrices.

Require: $m \geq 2$; an $(n-m+2) \times n$ matrix $\mathbf{V}_k^{(m-1)}$; a candidate $\alpha^{(m-1)}$ marked for elimination.
Ensure: $\alpha^{(m-1)}$ is removed from $\mathbf{V}_k^{(m-1)}$ to create $\mathbf{V}_k^{(m)}$.

```
procedure SHIFT(α^(m−1), V_k^(m−1))
    for l ∈ [1, n − m + 2] do
        if (v_l)_k^(m−1) encodes α^(m−1) then
            for i ∈ [l, n − m + 1] do
                // Shift up all choices below an eliminated row.
                (v_i)_k^(m−1) ← (v_{i+1})_k^(m−1)
    // Remove the eliminated row or any remaining duplicate.
    delete (v_{n−m+2})_k^(m−1)
```

2.3 Tie-Breaking Eliminations

It is possible that multiple candidates in a round may jointly qualify for elimination; in this case, we have a tie that must be broken. There are various methods to break ties, each with their own advantages and disadvantages. One method is to simply choose at random a candidate in the tie to eliminate. This method is supported by Lundell, who argues that the exclusion of chance in a tie-breaker encourages voters to be insincere when listing their preferences [19]. Another method is to use information from earlier rounds to break any ties. O'Neill considers both forwards and backwards tie-breakers; forwards tie-breaking eliminates the candidate with the fewest preferences from the earliest round, whilst backwards tie-breaking eliminates the candidate with the fewest preferences from the latest round [20]. The latter method is favoured by O'Neill, arguing that the most relevant information to break a tie comes from the previous round, not the first round. It is also possible that a tie still happens in forward or backward tie-breaking; in this case, random tie-breaking may be used instead.

Both forwards and backwards tie-breaking methods are simple to implement within our system. Suppose a set $\mathcal{S}$ of candidates are tied for the fewest preferences at a round m. One must only look at the tallies $\mathbf{t}^{(m')}$ where $m' < m$ and then mark the first candidate $c \in \mathcal{S}$ that minimises $t_c^{(m')}$ for elimination. This candidate may then be eliminated using Algorithm 1 as normal. One must look at all tallies in ascending order of m' if using forwards tie-breaking, and conversely in descending order of m' if using backwards tie-breaking. If there are still ties in all $\mathbf{t}^{(m')}$, then the candidate to be eliminated is chosen by random by using a pre-agreed procedure.

Another approach suggested by Robert et al. is eliminating all tied candidates at once [21]. This avoids the need to look at information from prior rounds or choose any candidates at random to eliminate. The method we described in §2.2 only supports eliminating one candidate per round, however it can be adjusted to mimic multiple eliminations as follows. First, during the setup procedure, initialise an empty stack of candidates. Then, during a tallying round m, suppose $n_\alpha^{(m)}$ candidates have the fewest votes at the end of the round. Push all $n_\alpha^{(m)}$ candidates onto the stack and proceed to the next round. In the next round, pop one candidate from the stack and eliminate them as normal. Then, at the end of the round, only if the stack is empty, determine a new number $n_\alpha^{(m+1)}$ of candidates to eliminate and push them onto the stack. Repeat until all rounds are completed or a candidate is elected with more than 50% of the votes. This modification mimics eliminating multiple candidates at once by instead splitting the eliminations over multiple rounds, ensuring that no new candidate is selected for elimination until all candidates on the stack have been eliminated.

In §3, we describe how we use our method from §2.2 to construct an E2E verifiable IRV e-voting protocol. The cryptographic description of the e-voting protocol remains the same, regardless of which tie-breaking method is chosen.

3 THE CAMEL SYSTEM

We now present Camel, an E2E verifiable system for IRV without requiring any tallying authorities. Our description of Camel is in the context of onsite voting in a polling station.

3.1 Requirements and Assumptions

We require that only eligible voters can vote and their eligibility is checked by election staff at the polling station. At the polling station, voters are authenticated with identifying documents before being given an anonymous voting credential (e.g., a random passcode or a smart card). With this credential, the voter enters a private voting booth and logs onto a Direct Recording Electronic (DRE) machine with a touch-screen interface, which allows the voter to rank the candidates in an order of their liking. The DRE machine records the voter's ranked choice in a ballot but does not know the voter's real identity, i.e., voting is anonymous. Camel only supports full rankings of candidates and we leave support for partial or incomplete rankings for future work.

We assume multiple DRE machines are securely connected to a server, which could be local in a polling station or placed at a precinct level. We regard the front-end DRE machines and the server together as one Camel system. We assume an external public bulletin board, where the Camel system has append-only write access (over a secure TLS channel) and everyone from the public has read access. By the nature of a touch-screen voting interface, the system learns a voter's choice by definition, but the voter's privacy is still preserved through anonymity. The plaintext ballots are encrypted in an election; the encrypted ballots, in conjunction with their well-formedness proofs, are published as the election progresses. After the election, additional zero-knowledge proofs are published to allow anyone to publicly verify the elimination of the least-preferred candidate and the transfer of votes in each IRV round without revealing any secret information about the vote.

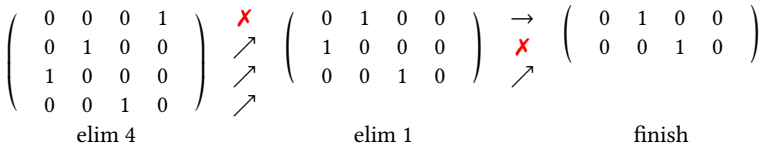

Figure 1: Eliminating two candidates 4 and 1 from the permutation matrix of $(4, 2, 1, 3)$. Here, 4 is eliminated in the first round and 1 is eliminated in the second round. The result is a shifted matrix of size 2×4.

3.2 Cryptographic Description

The Camel system follows the same method as given in §2.2. We now explain how cryptographic techniques are applied to this method to create a verifiable IRV voting system needing no TAs.

3.2.1 Setup. We initialise a mapping of each candidate in C to $[1, n]$ and the vectors $\mathbf{t}^{(m)}$ ($m \in [1, n-1]$) as described in §2.2. Additionally, let p and q be two large primes such that $q \mid p-1$. Denote by $\mathbb{G}_q$ the subgroup (of prime order q) of the group $\mathbb{Z}_p^*$. Unless stated otherwise, we assume all modular operations to be performed under the modulus p. We also require two random generators of $\mathbb{G}_q$, which we denote g_0 and g_1. To construct g_0 and g_1, we may fix g_0 to be a non-identity element in $\mathbb{G}_q$ and compute g_1 based on a one-way hash function including g_0 and public contextual information about the election (e.g., election name, date and candidates) in the input [22]. This ensures that the discrete logarithm relationship between g_0 and g_1, namely, $\log_{g_0} g_1$, is unknown to anyone. Finally, we initialise additional vectors $\mathbf{s}^{(m)} \leftarrow \left(s_1^{(m)}, \ldots, s_n^{(m)}\right)$ with $s_i^{(m)} = 0$ ($i \in [1, n]$) per each round $m \in [1, n-1]$. The purpose of $\mathbf{s}^{(m)}$ is to hold the sum of all randomness generated for the most preferred choice in a vote at round m. This is necessary for the final verification of the tally.

3.2.2 Vote Casting Phase. The voter only needs to interact with the DRE machine once during the Vote Casting phase. The voter submits their ranking of n candidates in their order of preference. The vote is transformed into a permutation matrix $\mathbf{V}_k$ for the voter k. The DRE server then generates a matrix $\mathbf{X}_k = (x_{ij})_k$ of random factors where $(x_{ij})_k \in_R \mathbb{Z}_q^*$ and computes the ballot $(B_{ij})_k = \langle (b_{ij})_k, (Y_{ij})_k \rangle$ as follows.

$$(b_{ij})_k \leftarrow g_0^{(x_{ij})_k} g_1^{(v_{ij})_k} \tag{BC_1}$$

$$(Y_{ij})_k \leftarrow g_1^{(x_{ij})_k} \tag{BC_2}$$

Here, we use "BC" to denote *ballot construction*. (BC_1) and (BC_2) create ElGamal ciphertexts fulfilling the following logical relation (which can be enforced by a disjunctive ZKP [23]).

$$\left(\log_{g_0}(b_{ij})_k / g_1 = \log_{g_1}(Y_{ij})_k\right) \vee \left(\log_{g_0}(b_{ij})_k = \log_{g_1}(Y_{ij})_k\right)$$

The DRE machine constructs a set of non-interactive ZKPs (NIZKPs) proving well-formedness of the ballots. We discuss the details of these in §3.3. The voter's ballot $(B_{ij})_k$, alongside its corresponding NIZKPs, is printed onto a receipt together with a digital signature to prove authenticity.

To facilitate the *cast as intended* property, we use a similar voter-initiated auditing procedure proposed by Benaloh [24]. The voter may *audit* or *confirm* their ballot. In the case of auditing the ballot, the DRE machine adds the voter's ranking of candidates and the matrix $\mathbf{X}_k$ to the printed receipt. The corresponding ballot $(B_{ij})_k$ is included in a set $\mathbb{A}$ of audited ballots and the entire receipt is posted to the public bulletin board. The voter may then check that the ranking on the receipt is consistent with their chosen order; if not, a dispute should be raised with the election organisers immediately. Should a voter opt to confirm their ballot, then the ballot is included in a set $\mathbb{C}$ of confirmed ballots and the DRE machine prints a ballot confirmation message on the receipt together with a digital signature. The same receipt is also published to the bulletin board, which the voter can check afterwards.

3.2.3 Tallying Phase. Once the voting server has received all votes, the Tallying phase begins. This phase operates over multiple rounds. During a round m, the voting server first eliminates the candidate $\alpha^{(m-1)}$ marked from the previous round from all matrices $\mathbf{V}_k^{(m-1)}$ using Algorithm 1 (or simply sets $\mathbf{V}_k^{(1)} = \mathbf{V}_k$ if $m = 1$) to create $\mathbf{V}_k^{(m)}$. The server then computes the round-based ballots $(B_{ij})_k^{(m)}$ by performing a new encryption of $\mathbf{V}_k^{(m)}$ using (BC_1) and (BC_2). This re-encryption is a necessary step for ensuring the well-formedness of ballots, and as such the voting server computes an additional NIZKP of well-formedness during each round. In short, these proofs show that the matrix $\mathbf{V}_k^{(m)}$ is a modification of $\mathbf{V}_k^{(m-1)}$ with only the row encoding the eliminated candidate removed. We discuss these in further detail in §3.3. Figure 2 illustrates the re-encryption process of the Tallying phase for a 5-candidate election.

Once all matrices $\mathbf{V}_k^{(m)}$ and their corresponding well-formedness proofs have been constructed for each voter, the voting server then updates the vectors $\mathbf{t}^{(m)}$ and $\mathbf{s}^{(m)}$ as follows.

$$\mathbf{t}^{(m)} \leftarrow \sum_{c \in \mathbb{C}} (\mathbf{v}_1)_c^{(m)} \tag{T_1}$$

$$\mathbf{s}^{(m)} \leftarrow \sum_{c \in \mathbb{C}} (\mathbf{x}_1)_c^{(m)} \tag{T_2}$$

Here, $(\mathbf{v}_1)_c^{(m)}$ and $(\mathbf{x}_1)_c^{(m)}$ simply denote the first rows of the matrix $\mathbf{V}_c^{(m)}$ and randomness $\mathbf{X}_c^{(m)}$ generated at a round m respectively for a confirmed ballot. The randomness must be kept secret until the end of the round, at which point it must be securely deleted. The tally $\mathbf{t}^{(m)}$ and the vector $\mathbf{s}^{(m)}$ are published to the public bulletin board and $\mathbf{t}^{(m)}$ is directly inspected to determine if there is a winner or if there is a least preferred candidate $\alpha^{(m)}$ to mark and eliminate at the start of the next round. The new ballot $(B_{ij})_k^{(m)}$ is appended to the voter's receipt on the bulletin board (and also signed), allowing them to check the receipt and verify that

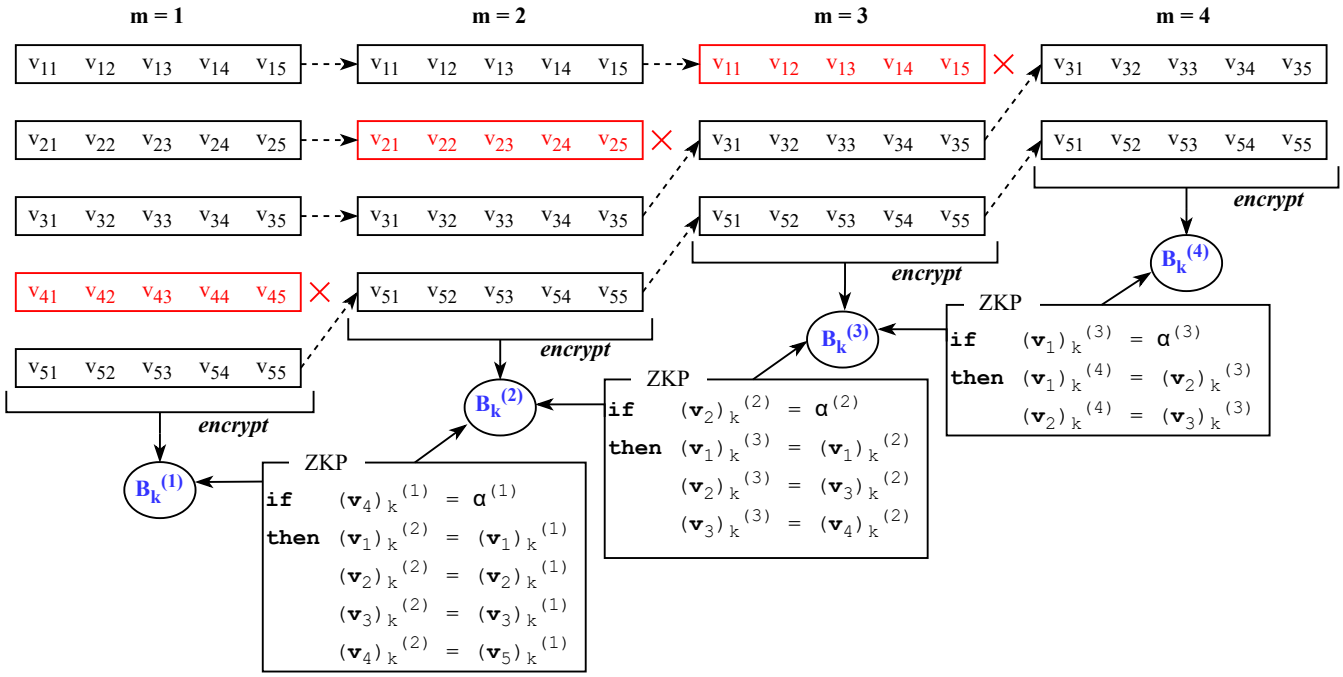

Figure 2: The shifting and re-encryption processes for the Tallying phase. Here, $(\mathbf{v}_i)_k^{(m)}$ denotes the row i of matrix $\mathbf{V}_k^{(m)}$ at round m. For the first round $m = 1$, the matrix $\mathbf{V}_k^{(1)}$ is taken to be $\mathbf{V}_k$ from the Vote Casting phase. $\mathbf{V}_k^{(1)}$ is encrypted to create $B_k^{(1)}$. During all other rounds $m \geq 2$, one row is removed from the matrix $\mathbf{V}_k^{(m-1)}$ (the one encoding the candidate $\alpha^{(m-1)}$ marked for elimination) using Algorithm 1. This creates a new matrix $\mathbf{V}_k^{(m)}$ for the next round. The matrix $\mathbf{V}_k^{(m)}$ is encrypted to create the ballot $B_k^{(m)}$. Then, NIZKPs of well-formedness are constructed between the ballots $B_k^{(m-1)}$ and $B_k^{(m)}$, proving that $\mathbf{V}_k^{(m)}$ is the result of removing the row encoding $\alpha^{(m-1)}$ from $\mathbf{V}_k^{(m-1)}$. The voting server computes the tallies $\mathbf{s}^{(m)}$ and $\mathbf{t}^{(m)}$ and marks a new candidate $\alpha^{(m)}$ to eliminate during the next round. The whole procedure repeats until a winner may be determined.

all eliminations are being performed correctly by Camel. Camel then moves onto a new round and repeats this phase until a winner is elected.

3.2.4 Tally Verification. Once the Tallying phase ends and the election is complete, anyone may perform the following checks to verify the integrity of the tallies at each round. First, the NIZKPs of well-formedness hold. Second, the digital signatures are valid. Third, the following equations hold for all $m \in [1, n-1]$, $j \in [1, n]$.

$$\prod_{c \in \mathbb{C}} (b_{1j})_c^{(m)} = g_0^{s_j^{(m)}} g_1^{t_j^{(m)}} \tag{TV_1}$$

$$\prod_{c \in \mathbb{C}} (Y_{1j})_c^{(m)} = g_1^{s_j^{(m)}} \tag{TV_2}$$

We use "TV" to denote *tally verification*. Anyone with access to the public bulletin board can verify these equations.

3.3 Ballot Well-formedness

NIZKPs are necessary to ensure the well-formedness of all cast ballots in Camel. Ballots are encrypted once during the Vote Casting phase, and are then re-encrypted in all subsequent rounds in the Tallying phase. As such, we consider the well-formedness proofs for the Vote Casting phase and the Tallying phase separately.

3.3.1 Well-formedness during Vote Casting. In the Vote Casting phase, ballots are constructed from permutation matrices; hence we define a ballot to be *well-formed during Vote Casting* if its underlying plaintext vote is a permutation matrix. We may use the following Theorem to determine whether a binary matrix is also a permutation matrix.

Theorem 3.1. An $n \times n$ binary matrix $\mathbf{V}_k = (v_{ij})_k$ satisfying $(v_{ij})_k = 0 \vee (v_{ij})_k = 1$ is a permutation matrix if and only if $\sum_{i=1}^{n}(v_{ij})_k = \sum_{j=1}^{n}(v_{ij})_k = 1$.

Proof. ($\Rightarrow$). Suppose $\mathbf{V}_k$ is a permutation matrix. Then each row of $\mathbf{V}_k$ is an encoding of a different candidate from C. Since each encoding only contains a single 1, it is trivial that $\sum_{i=1}^{n}(v_{ij})_k = 1$. Additionally, since each encoding is for a different candidate, each column of $\mathbf{V}_k$ contains only a single 1 with all other values at 0, hence $\sum_{j=1}^{n}(v_{ij})_k = 1$.

($\Leftarrow$). Suppose $\sum_{i=1}^{n}(v_{ij})_k = \sum_{j=1}^{n}(v_{ij})_k = 1$. Since $\mathbf{V}_k$ is a binary matrix, there must be only one 1 in each row for $\sum_{i=1}^{n}(v_{ij})_k = 1$ to

hold, hence each row is an encoding of a candidate from C. Additionally, since $\sum_{j=1}^{n}(v_{ij})_k = 1$, these entries of 1 must be in different columns, hence each encoding is for a unique candidate from C. Therefore $\mathbf{V}_k$ is a collection of unique encodings of candidates from C and is, by definition, a permutation matrix. □

We use Theorem 3.1 to construct the following NIZKP of well-formedness P_{WF}^{VC} for ballots created during the Vote Casting phase.

$$
\begin{aligned}
&P_{WF}^{VC}\left\{(B_{ij})_k = \left\langle (b_{ij})_k, (Y_{ij})_k \right\rangle\right\} = \\
&P_K\Bigg\{(x_{ij})_k : \\
&\quad \Big(\Big(\log_{g_0}(b_{ij})_k / g_1 = \log_{g_1}(Y_{ij})_k\Big) \vee \Big(\log_{g_0}(b_{ij})_k = \log_{g_1}(Y_{ij})_k\Big)\Big) \\
&\quad \wedge \left(\log_{g_0}\left(\prod_{i=1}^{n}(b_{ij})_k\right)/g_1 = \log_{g_1}\prod_{i=1}^{n}(Y_{ij})_k\right) \\
&\quad \wedge \left(\log_{g_0}\left(\prod_{j=1}^{n}(b_{ij})_k\right)/g_1 = \log_{g_1}\prod_{j=1}^{n}(Y_{ij})_k\right) \Bigg\}
\end{aligned}
$$

The first statement ensures that $\mathbf{V}_k$ is a binary matrix. The last two conjunctive statements result from utilising Theorem 3.1 alongside the additively homomorphic property of the ElGamal ciphertexts. P_{WF}^{VC} should hold for all $i, j \in [1, n]$.

3.3.2 Well-formedness during Tallying. We require an additional theorem of well-formedness for the ballots that are re-encrypted during the Tallying phase. This is because rows of the matrices are deleted between rounds in this phase. We define a matrix $\mathbf{V}_k^{(m)}$ to be *well-formed during Tallying* if it is the result of shifting the matrix $\mathbf{V}_k^{(m-1)}$ using Algorithm 1 with $\alpha^{(m-1)}$ from the previous round.

Consider first the row l of $\mathbf{V}_k^{(m-1)}$ that encodes $\alpha^{(m-1)}$. By definition of Algorithm 1, only the rows $i > l$ of $\mathbf{V}_k^{(m-1)}$ are shifted upwards by one index to create $\mathbf{V}_k^{(m)}$. All other rows $i < l$ remain the same between $\mathbf{V}_k^{(m-1)}$ and $\mathbf{V}_k^{(m)}$. This means we can verify the well-formedness of a matrix $\mathbf{V}_k^{(m)}$ with the following expression.

$$
\begin{aligned}
&\text{if} \quad (v_{l\alpha^{(m-1)}})_k^{(m-1)} = 1 \wedge \textstyle\sum_{j\neq\alpha^{(m-1)}}(v_{lj})_k^{(m-1)} = 0 \\
&\text{then} \quad \textstyle\bigwedge_{i\in[1,l-1]}(\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_i)_k^{(m-1)} \\
&\qquad \wedge \textstyle\bigwedge_{i\in[l,n-m+1]}(\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_{i+1})_k^{(m-1)}
\end{aligned}
$$

Expression 1: Verification of a row between $\mathbf{V}_k^{(m-1)}$ and $\mathbf{V}_k^{(m)}$.

This expression first checks that the bit at index $\alpha^{(m-1)}$ of row l is equal to 1, with all other bits of the row equal to 0, i.e., that row l is an encoding of $\alpha^{(m-1)}$. If this holds, then all rows $i < l$ must remain the same between $\mathbf{V}_k^{(m-1)}$ and $\mathbf{V}_k^{(m)}$, whilst all other rows $i > l$ of $\mathbf{V}_k^{(m-1)}$ must have been shifted upwards by one place to create $\mathbf{V}_k^{(m)}$. Examples are illustrated in Figure 2.

We wish to apply Expression 1 over all rows of the matrices $\mathbf{V}_k^{(m-1)}$ and $\mathbf{V}_k^{(m)}$. To achieve this, note that only one row of $\mathbf{V}_k^{(m-1)}$ may encode $\alpha^{(m-1)}$. Therefore, we may rewrite Expression 1 as a 1-of-$(n-m+2)$ disjunctive statement. This is stated by the following Theorem.

Theorem 3.2. An $(n-m+2)\times n$ binary matrix $\mathbf{V}_k^{(m)}$ $(m \geq 2)$ is well-formed during round m if and only if the following holds.

$$
\bigvee_{l\in[1,n-m+2]}\left(\gamma \wedge \left(\bigwedge_{i\in[1,l-1]}\psi\right) \wedge \left(\bigwedge_{i\in[l,n-m-1]}\psi'\right)\right)
$$

Where we have:

- $\gamma \leftarrow (v_{l\alpha^{(m-1)}})_k^{(m-1)} = 1 \wedge \sum_{j\neq\alpha^{(m-1)}}(v_{lj})_k^{(m-1)} = 0$
- $\psi \leftarrow (\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_i)_k^{(m-1)}$
- $\psi' \leftarrow (\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_{i+1})_k^{(m-1)}$

With $\alpha^{(m-1)}$ as the candidate marked for elimination from the previous round $m-1$.

Proof. Follows from definition of Algorithm 1. □

The statement γ verifies that a row l of $\mathbf{V}_k^{(m-1)}$ encodes $\alpha^{(m-1)}$; if it does, then the statements ψ and ψ' ensure that all rows $i < l$ are the same between $\mathbf{V}_k^{(m-1)}$ and $\mathbf{V}_k^{(m)}$, whilst all other rows $i > l$ of $\mathbf{V}_k^{(m-1)}$ were shifted upwards by one place to create $\mathbf{V}_k^{(m)}$. Note that Theorem 3.2 only applies for rounds $m \geq 2$, as two successive round-based ballots are needed. For the first round, there will trivially be no candidates to eliminate and we have $\mathbf{V}_k^{(1)} = \mathbf{V}_k$, so Theorem 3.1 and P_{WF}^{VC} are sufficient to rely on for $m = 1$ only.

Theorem 3.2 uses only conjunctive and disjunctive statements and propositions of equality. This means we may use the standard techniques of conjunctive and disjunctive knowledge [23] to prove Theorem 3.2 holds between two encrypted ballots $B_k^{(m-1)}$ and $B_k^{(m)}$. We refer to this proof of well-formedness as P_{WF}^{T} for the Tallying phase. We denote the direct translations of γ, ψ and ψ' into zero knowledge statements as Γ, Ψ and Ψ' respectively. The full definitions of Γ, Ψ and Ψ' are given in Appendix A.

4 SECURITY ANALYSIS

4.1 E2E Verifiability

We discuss that Camel satisfies the three requirements of E2E verifiability, namely the *cast as intended*, *recorded as cast* and *tallied as recorded* requirements [10]. Since we follow the Benaloh vote casting approach [24], it is straightforward to see that Camel satisfies the "cast as intended" requirement. Camel commits to encrypted ballots by appending them onto a receipt. When a voter chooses to audit their ballot, the randomness $\mathbf{X}_k$ and the voter's plaintext matrix $\mathbf{V}_k$ are revealed on the receipt. The voter may check that $\mathbf{V}_k$ is the permutation matrix for their intended vote. The receipt is also published on the bulletin board, so anyone may verify that the ciphertext is the correct encryption of $\mathbf{V}_k$ using randomness $\mathbf{X}_k$. Also, the tallying rounds are universally verifiable – provided that the well-formedness verifications hold, and the voter's initial ranking is recorded correctly, then the voter can be sure that their vote is being transformed correctly by Camel through Algorithm 1.

Voters may choose to audit their vote an unpredictable number of times. A voter may choose to confirm their vote once they are convinced their vote is being cast correctly. The voter should check their receipt then matches the one held on the bulletin board. If both receipts match, then the voter can be sure that Camel has recorded their vote, fulfilling the "recorded as cast" requirement.

The following theorem shows that Camel satisfies the "tallied as recorded" requirement, provided that the proofs of well-formedness and the tally verification equations hold over all rounds.

Theorem 4.1. In Camel, assuming the proofs of well-formedness P^{VC}_{WF} and P^{T}_{WF} hold, and the tally verification equations (TV_1) and (TV_2) also hold, then the reported tallies $\mathbf{t}^{(m)}$ are the correct tallies of all confirmed votes on the bulletin board per tallying round m. That is, the tally at the round m is the correct tally of all voters' most preferred candidates as required for an IRV election.

Proof. We show that the proofs of well-formedness P^{VC}_{WF} and P^{T}_{WF}, and the tally verification equation (TV_2), collectively guarantee that the remaining tally verification equation (TV_1) holds iff $\mathbf{t}^{(m)} = \sum_{c \in \mathbb{C}} \mathbf{v}_1^{(m)}$ for each tallying round m.

($\Rightarrow$). Suppose (TV_1) holds and $\prod_{c \in \mathbb{C}} (b_{1j})_c^{(m)} = g_0^{s_j^{(m)}} g_1^{t_j^{(m)}}$. Through definition of $(B_{ij})_k$, we have $(b_{1j})_k^{(m)} = g_0^{(x_{1j})_k^{(m)}} g_1^{(v_{1j})_k^{(m)}}$ and therefore the following holds.

$$\prod_{c \in \mathbb{C}} (b_{1j})_c^{(m)} = \prod_{c \in \mathbb{C}} g_0^{(x_{1j})_c^{(m)}} g_1^{(v_{1j})_c^{(m)}} = g_0^{\sum_{c \in \mathbb{C}} (x_{1j})_c^{(m)}} g_1^{\sum_{c \in \mathbb{C}} (v_{1j})_c^{(m)}}$$

It is clear that $\mathbf{s}^{(m)} = \sum_{c \in \mathbb{C}} (\mathbf{x}_1)_c^{(m)}$ and $\mathbf{t}^{(m)} = \sum_{c \in \mathbb{C}} (\mathbf{v}_1)_c^{(m)}$.

($\Leftarrow$). Suppose $\mathbf{t}^{(m)} = \sum_{c \in \mathbb{C}} (\mathbf{v}_1)_c^{(m)}$. By definition of $(B_{ij})_k$, we have $\prod_{c \in \mathbb{C}} (b_{1j})_c^{(m)} = g_0^{\sum_{c \in \mathbb{C}} (x_{1j})_c^{(m)}} g_1^{\sum_{c \in \mathbb{C}} (v_{1j})_c^{(m)}}$. Then, by applying ($TV_2$), we have the following.

$$\prod_{c \in \mathbb{C}} (Y_{1j})_k^{(m)} = g_1^{\sum_{c \in \mathbb{C}} (x_{1j})_c^{(m)}} = g_1^{s_j^{(m)}}$$

Hence $\mathbf{s}^{(m)} = \sum_{c \in \mathbb{C}} (\mathbf{x}_1)_c^{(m)}$. By substituting $s_j^{(m)}$ and $t_j^{(m)}$ into $\prod_{c \in \mathbb{C}} (b_{1j})_c^{(m)}$, we receive $g_0^{s_j^{(m)}} g_1^{t_j^{(m)}}$. This is precisely ($TV_1$). □

Therefore, provided the well-formedness and tally verifications are performed successfully, the reported tallies $\mathbf{t}^{(m)}$ are the correct tallies, for each tallying round m, of all of the confirmed votes on the bulletin board in Camel.

4.2 Ballot Secrecy

We now consider *ballot secrecy* for an election, which describes the principle that any voting system should preserve the privacy of any individual's vote. For our work on Camel, we make use of Benaloh's definition of vote privacy [25], and define it to be maintained if an attacker colluding with η dishonest voters has only a negligible chance to distinguish between any two elections having the same partial tally of honest votes. We note that, in 2015, Bernhard et al. [26] proposed BPRIV, which is an alternative game-based definition of privacy. However, we favour Benaloh's definition for our work. This is because BPRIV assumes an "honest single trustee" (which can be extended to multiple trustees), that not only performs decryption, but is also tasked with removing any duplicated ballots. However, Camel does not involve any TAs (trustees) by design. We first state the Decision Diffie-Hellman (DDH) assumption [27] and assume it to be hard in $\mathbb{G}_q$.

Assumption 4.1. Given g, g^a, g^b and $\Omega \in \{g^{ab}, R\}$, where $a, b \in \mathbb{Z}_q^*$ and $R \in \mathbb{G}_q$, it is hard to decide whether $\Omega = g^{ab}$ or $\Omega = R$.

We then require the following assumptions. We state these explicitly and show they are implied from the DDH assumption.

Assumption 4.2. Consider the following security experiment $Exp^{RND}_{\mathcal{A}}(\lambda)$. For any $g^a, g^b \in \mathbb{G}_q$, define $DH_g(g^a, g^b) = g^{ab}$.

$Exp^{RND}_{\mathcal{A}}(\lambda)$
$g \xleftarrow{\$} \mathbb{G}_q$
$A \xleftarrow{\$} \mathbb{G}_q$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}^{O(\cdot)}(g, A)$
Return $d = d'$

$O()$
$B \xleftarrow{\$} \mathbb{G}_q$
$\Omega_0 \leftarrow DH_g(A, B)$
$\Omega_1 \xleftarrow{\$} \mathbb{G}_q$
Return (B, Ω_d)

In this experiment, the challenger first selects, at random, two elements g and A from $\mathbb{G}_q$. It then invokes the adversary $\mathcal{A}$ with these elements. $\mathcal{A}$ has access to the oracle O and may query it $poly(\lambda)$ times. On every query, O selects a random element B from $\mathbb{G}_q$ and computes $DH_g(A, B)$. O then returns B, and either $DH_g(A, B)$ or a new random element from $\mathbb{G}_q$; the choice depends on a secret bit d chosen by the challenger. The task of the adversary is to guess d.

The advantage of an adversary $\mathcal{A}$, against the security experiment $Exp^{RND}_{\mathcal{A}}(\lambda)$, is defined as below.

$$Adv^{RND}_{\mathcal{A}}(\lambda) = \left| Pr[Exp^{RND}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

For any PPT adversary $\mathcal{A}$, $Adv^{RND}_{\mathcal{A}}(\lambda) \leq negl(\lambda)$.

Lemma 4.1. *The DDH assumption implies assumption 4.2.*

Proof. Proved as Lemma 4 in Kurosawa and Nojima [28]. □

Assumption 4.3. Consider the following security experiment $Exp^{RND1}_{\mathcal{A}}(\lambda)$. The adversary passes two inputs to the oracle O on each query. Both of the inputs are a single bit. O randomly selects one of them in correspondence with another secret bit d chosen by the challenger, and uses the chosen value to compute Ω_d. O then returns B and Ω_d to the adversary. Everything else is the same as the security experiment $Exp^{RND}_{\mathcal{A}}(\lambda)$.

$Exp^{RND1}_{\mathcal{A}}(\lambda)$
$g \xleftarrow{\$} \mathbb{G}_q$
$A \xleftarrow{\$} \mathbb{G}_q$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(g, A)$
Return $d = d'$

$O(v_0, v_1)$
$B \xleftarrow{\$} \mathbb{G}_q$
$\Omega_0 \leftarrow DH_g(A, B) * g^{v_0}$
$\Omega_1 \leftarrow DH_g(A, B) * g^{v_1}$
Return (B, Ω_d)

The advantage of an adversary $\mathcal{A}$, against the security experiment $Exp^{RND1}_{\mathcal{A}}(\lambda)$, is then given as the following.

$$Adv^{RND1}_{\mathcal{A}}(\lambda) = \left| Pr[Exp^{RND1}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

For any PPT adversary $\mathcal{A}$, $Adv^{RND1}_{\mathcal{A}}(\lambda) \leq negl(\lambda)$.

Lemma 4.2. *Assumption 4.2 implies 4.3.*

Proof. The lemma can be easily proven by applying the triangle inequality for computational indistinguishability [29]. This states that $SD(D_0, D_2) \leq SD(D_0, D_1) + SD(D_1, D_2)$ is true for any three distributions D_0, D_1, and D_2, where SD denotes statistical difference. Using the triangle inequality, one can show that the inequality $Adv^{RND1}_{\mathcal{A}}(\lambda) \leq 2 * Adv^{RND}_{\mathcal{A}}(\lambda)$ holds. □

We now define the indistinguishability notion in an IRV election. The adversary must choose two different sets of votes in a tallying round m having the same cardinality and the same tally. The challenger randomly chooses one of them and converts the set of votes into encrypted ballots using (BC_1) and (BC_2). The adversary's task is to identify the set of votes selected by the challenger. The only condition is that the tallies must be equal (it is easy to see that if the tallies in the two bulletin boards are unequal, the adversary can trivially distinguish between the two bulletin boards). This is formalised through the following definition and subsequent lemma.

Definition 4.3. Let us consider the following security experiment $Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$. In this experiment, the challenger first chooses two random generators g_0 and g_1. It then creates two bulletin boards BB_0, and BB_1, both initially empty. It also stores three variables X,V_0, and V_1 (of size $(n-m+1) \times n$) initialized with zeroes, i.e. $X(i,j) = V_0(i,j) = V_1(i,j) = 0, \forall i \in [1, n-m+1], j \in [1,n]$. Assume that Γ is the set of all well-formed $(n-m+1) \times n$ matrices at round m. The challenger then invokes $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. $\mathcal{A}$ has access to the oracle O, and on each query, passes two inputs $v_0 \in \Gamma$ and $v_1 \in \Gamma$ to O. O then selects a random $(n-m+1) \times n$ matrix $x \in \mathbb{Z}_p^{n\cdot(n-m+1)}$, and generates the variables c_0 and c_1 as shown in the experiment; these represent the ballots for the votes v_0 and v_1. It stores c_0 and c_1 in the bulletin boards BB_0 and BB_1 respectively. It also stores the cumulative values of the selected randomnesses in the matrix X. Similarly, it stores the cumulative values of v_0, and v_1 in variables T_0 and T_1 respectively.

When $\mathcal{A}_0$ returns, the challenger then invokes $\mathcal{A}_1$ with X and one of BB_0 or BB_1. The goal of $\mathcal{A}_1$ is to identify the correct bulletin board selected by the challenger. $\mathcal{A}$ wins the game if $\mathcal{A}_1$ can identify the bulletin board, provided the tallies in both of them are the same.

$Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$
$g_1, g_0 \xleftarrow{\$} \mathbb{G}_q$
$BB_0 = BB_1 = \emptyset$
$V_0 = V_1 = 0$
$X = 0$
$st \leftarrow \mathcal{A}_0^{O(\cdot,\cdot)}(v_1, v_0)$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}_1(st, BB_d, X)$
Return $(T_0 = T_1) \wedge (d = d')$

$O(v_0, v_1)$
$x \xleftarrow{\$} \mathbb{Z}_p^{n\cdot(n-m+1)}$
$c_0(i,j) \leftarrow (g_0^{x(i,j)} g_1^{v_0(i,j)}, g_1^{x(i,j)}) : i \in [1, n-m+1], j \in [1,n]$
$c_1(i,j) \leftarrow (g_0^{x(i,j)} g_1^{v_1(i,j)}, g_1^{x(i,j)}) : i \in [1, n-m+1], j \in [1,n]$
$X(i,j) \leftarrow X(i,j) + x(i,j) : i \in [1, n-m+1], j \in [1,n]$
$BB_i \leftarrow BB_i \bigcup \{c_i\} : i = 0, 1$
$T_0(i,j) \leftarrow T_0(i,j) + v_0(i,j) : i \in [1, n-m+1], j \in [1,n]$
$T_1(i,j) \leftarrow T_1(i,j) + v_1(i,j) : i \in [1, n-m+1], j \in [1,n]$

The advantage of an adversary $\mathcal{A}$, against the security experiment $Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$ is defined as below.

$$Adv^{IND-Vote}_{\mathcal{A}}(\lambda) = \left| Pr[Exp^{IND-Vote}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|$$

Lemma 4.4. *For any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, it holds that $Adv^{IND-Vote}_{\mathcal{A}}(\lambda) \leq negl(\lambda)$.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B}$, against the security experiment $Exp^{RND1}_{\mathcal{B}}(\lambda)$ of Assumption 4.3. $\mathcal{B}$ works as follows. It first receives as input g and A according to Assumption 4.3. $\mathcal{B}$ then invokes $\mathcal{A}$ with g and A, i.e., $g_1 = g$ and $g_0 = A$. On each query that $\mathcal{A}_0$ makes to O with input v_0 and v_1, the adversary $\mathcal{B}$ makes $n \cdot (n-m+1)$ queries of the form $(v_{0t}(i,j), v_{1t}(i,j))$ in the security experiment $Exp^{RND1}_{\mathcal{B}}(\lambda)$, where $i \in [1, n-m+1]$, $j \in [1,n]$, and $t \in [1, \eta]$. Here, η denotes the total number of queries to O made by $\mathcal{A}_0$ in $Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$. $\mathcal{B}$ does not know the value of η; it makes a random guess of $\eta \in_R poly(\lambda)$. If the guess is incorrect, $\mathcal{B}$ aborts and returns a random bit. $\mathcal{B}$ receives $(B(i,j), \Omega_d(i,j))$ as the value returned by the oracle in $Exp^{RND1}_{\mathcal{B}}(\lambda)$.

For each of the first $\eta - 1$ queries, $\mathcal{B}$ makes $n\cdot(n-m+1)$ queries to its internal oracle, and receives $n\cdot(n-m+1)$ responses which we give as $(B_k(i,j), \Omega_{dk}(i,j)), \forall i \in [1, n-m+1], j \in [1,n], k \in [1, \eta-1]$. The input for the last query made by $\mathcal{A}_0$ in $Exp^{IND-Vote}_{\mathcal{A}}(\lambda)$ is simply $(v_{0\eta}, v_{1\eta})$. Assign $V(i,j) = \sum_{k=1}^{\eta} v_{0k}(i,j) = \sum_{k=1}^{\eta} v_{1k}(i,j)$, $\forall i \in [1, n-m+1]$, $j \in [1,n]$. $\mathcal{B}$ selects random $X(i,j) \xleftarrow{\$} \mathbb{Z}_p$, $\forall i \in [1, n-m+1]$, $j \in [1,n]$, and assigns the values $B_\eta(i,j)$ and $\Omega_{d\eta}(i,j)$ as follows.

$$B_\eta(i,j) = \frac{A^{X(i,j)}}{\prod_{k=1}^{\eta-1} B_k(i,j)}$$

$$\Omega_{d\eta}(i,j) = \frac{g_0^{X(i,j)} g_1^{V(i,j)}}{\prod_{k=1}^{\eta-1} \Omega_{dk}(i,j)}$$

Where $i \in [1, n-m+1]$, $j \in [1,n]$, and $\log_{g_0} g_1$ is unknown. Denote $BB_d = \{(B_k, \Omega_{dk}) : k \in [1, \eta]\}$. $\mathcal{B}$ simulates all ZKPs and invokes $\mathcal{A}_1$ with BB_d. If $\mathcal{A}_1$ can identify d, so can $\mathcal{B}$. If $\mathcal{B}$ can correctly guess the value of η, then their advantage is the same as that of $\mathcal{A}$. Otherwise, if $\mathcal{B}$ cannot guess it correctly, then their advantage is 0. We may therefore write the following.

$$Pr[Exp^{RND1}_{\mathcal{B}}(\lambda) = 1]$$
$$\geq (1/poly(\lambda)) \cdot Exp^{IND-Vote}_{\mathcal{A}}(\lambda) + (1 - 1/poly(\lambda)) \cdot \frac{1}{2}$$

And hence:

$$Adv_{\mathcal{B}}^{RND1}(\lambda) \geq 1/poly(\lambda) \cdot Adv_{\mathcal{A}}^{IND-Vote}(\lambda)$$

From this, we get, $Adv_{\mathcal{A}}^{IND-Vote}(\lambda) \leq poly(\lambda) \cdot Adv_{\mathcal{B}}^{RND1}(\lambda)$. This completes the proof. □

Next we use experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ to prove our scheme is secure against an active adversary that can corrupt some but not all of the voters in an IRV election. In particular, we show that an attacker who corrupts an arbitrary number of voters will only learn the partial tally of the honest votes. The challenger selects between two sets of honest votes that have the same partial tally. The task of the adversary is to identify the set of votes selected by the challenger. All NIZKPs of well-formedness are simulated by the challenger; this means that the adversary has a negligible advantage in distinguishing the NIZKPs from real ones. The advantage of the adversary comes from the encrypted ballots. The following lemma proves that the attacker will have negligible advantage in determining the set of honest votes that was selected by the challenger.

Lemma 4.5. *Assume that in an arbitrary IRV election, there are v voters, where $v \in poly(\lambda)$. Denote by P_i the index for a voter $i \in [1, v]$. Let H be the set of indices of the honest voters. All other voters are corrupted by the adversary $\mathcal{A}$. Let V_i denote the matrix (at round m) that represents the vote of P_i. Consider two sets of matrices $\Theta_0 = \{\overrightarrow{V}_{0i} : i \in [1, |H|]\}$ and $\Theta_1 = \{\overrightarrow{V}_{1i} : i \in [1, |H|]\}$, satisfying $\sum_{i=1}^{|H|} \overrightarrow{V}_{0i} = \sum_{i=1}^{|H|} \overrightarrow{V}_{1i}$ (i.e., their tallies are the same). Then, the adversary cannot distinguish between the following two cases.*

(1) $\{V_i : i \in H\} = \Theta_0$.
(2) $\{V_i : i \in H\} = \Theta_1$.

Proof. We show that if there exists such an adversary $\mathcal{A}$, it could be used in the construction of another adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$, against the security experiment $Exp_{\mathcal{B}}^{IND-Vote}(\lambda)$. $\mathcal{B}$ functions as follows. First, $\mathcal{B}_0$ receives the two generators g_0 and g_1. Then, whenever $\mathcal{A}$ enters a vote v for a corrupt voter, $\mathcal{B}_0$ queries the oracle O with (v, v) (the two entries are the same). $\mathcal{B}_0$ then selects $\overrightarrow{V}_{0i}$ and $\overrightarrow{V}_{1i}$ for some $i \in [1, |H|]$, and queries O with $(\overrightarrow{V}_{0i}, \overrightarrow{V}_{1i})$.

As the number of honest voters is $|H|$, which is equal to $|\Theta_0|$ and $|\Theta_1|$, $\mathcal{B}_0$ can hence set a unique pair of votes from (Θ_0, Θ_1) that represents the vote of the i^{th} honest voter. $\mathcal{B}_0$ returns once all v queries to O are completed. Then, $\mathcal{B}_1$ is invoked with BB_d (for $d \in \{0, 1\}$) and sends it to $\mathcal{A}$. If $\mathcal{A}$ can identify d, so can $\mathcal{B}_1$. □

5 RELATED WORK

A number of verifiable e-voting schemes for IRV have been considered in the literature [11–15]. Of these, the most relevant to ours are the works of Benaloh et al. from 2009 [15] and Ramchen et al. from 2019 [11]. As such, we compare Camel against these two works.

Benaloh et al. proposed a verifiable voting protocol called *Shuffle-Sum* in 2009 [15]. This protocol implements *Single Transferable Vote* (STV), a more general voting protocol than IRV that allows for multiple winners, with the single-winner case reducing to IRV. Tallying in Shuffle-Sum uses homomorphic addition over first-preference ballots. Eliminations are handled through homomorphic subtraction over preference-order ballots, with eliminated candidates identified through encrypted *indicator* bits [15]. Their ballot representation schemes are more compact than ours, however the authors require a mixing scheme to convert between the first-preference and preference-order ballots, which is not necessary in Camel. The authors additionally require a set of *trustees* to compute the election result, with a threshold of these needing to be honest [15]. As Camel is free from any trustees, it avoids the added computational and communication costs between them, as well as any associated trust assumptions. The limitation of Camel compared to Shuffle-Sum is the lack of support for STV; we plan to explore support for STV as part of future work.

Ramchen et al. proposed a new MPC protocol designed for IRV in 2019 [11]. Their protocol has an additive property permitting tallying of votes in a source group, and a multiplicative property permitting the elimination of candidates in a target group. A verifiable encryption switching permits transformations between the two groups. The authors use a similar matrix representation of votes to ours, although the elimination procedure is different; candidates are eliminated by "striking out" their columns in the matrix. The tallying procedure is also different; their scheme produces an *indicator vector* describing which candidate a vote should count for using a homomorphic dot product. Their protocol follows the verifiability model of Schoenmakers and Veeningen [30] rather than E2E verifiability. This enables verification of the tally, although it is not possible for a voter to check that their vote is representative of their choice and that it is included in the tally. The authors also require a set of trustees to compute the election result. One limitation of Camel compared to the work of Ramchen et al. is that their scheme supports partial rankings of candidates [11], whilst Camel requires full rankings. This would require non-trivial modifications to our ballot representation scheme and well-formedness proofs; we leave this for future work.

6 CONCLUSION

In this paper, we proposed Camel: the first E2E verifiable system for IRV without tallying authorities. Camel utilises a matrix representation for votes and manages the elimination of candidates through a novel shifting procedure with universal verifiability. One candidate is eliminated in each tallying round, which permits verification through comparisons between the two ballots in any two successive rounds. We discussed the E2E verifiability of Camel and proved that it retains ballot secrecy and limits a colluding set of voters to learn nothing more than the partial tallies at each round.

Camel simplifies the election management in an E2E system by removing tallying authorities, which are present in related systems such as those of Benaloh et al. [15] and Ramchen et al. [11]. However, Camel has a limitation in that it currently only supports IRV (single-winner STV), while Benaloh et al.'s system can support multi-winner STV. Furthermore, Camel requires voters to give full rankings of all candidates, while Ramchen et al.'s scheme allows for partial rankings. We plan to study support for multi-winner STV and partial rankings in future work.

ACKNOWLEDGEMENT

We would like to acknowledge the support of EPSRC (EP/T014784/1).

REFERENCES

[1] R. Richie. Instant Runoff Voting: what Mexico (and others) could learn. *ELJ*, 3(3): 501–512, 2004.
[2] K. Tomlinson, J. Ugander, and J. Kleinberg. Ballot Length in Instant Runoff Voting. *arXiv preprint arXiv:2207.08958*, 2022.
[3] B.P. Marron. One Person, One Vote, Several Elections: Instant Runoff Voting and the Constitution. *Vermont Law Review*, 28:343, 2003.
[4] UK Labour Party. Labour Leadership and Deputy Leadership Elections 2020 - Procedure and timetable. 2020. URL https://labourlist.org/wp-content/uploads/2020/01/LE20-Procedures-and-Timetabale.pdf.
[5] A. Sen. Majority Decision and Condorcet Winners. *SC&W*, 54(2):211–217, 2020.
[6] R. Richie, J. Seitz-Brown, and L. Kaufman. The Case for Instant Runoff Voting. *CPE*, pages 1–11, 2023.
[7] A.Gibbard. Manipulation of Voting Schemes: a General Result. *Econometrica*, pages 587–601, 1973.
[8] M.A.Satterthwaite. Strategy-proofness and Arrow's Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions. *JET*, 10(2):187–217, 1975.
[9] J. Green-Armytage. Four Condorcet-Hare Hybrid Methods for Single-winner Elections. *VM*, 29:1–14, 2011.
[10] F. Hao and P.YA. Ryan. *Real-world Electronic Voting: Design, Analysis and Deployment.* CRC Press, 2016.
[11] K. Ramchen, C. Culnane, O. Pereira, and V. Teague. Universally Verifiable MPC and IRV Ballot Counting. In *FC 2019*, pages 301–319. Springer, 2019.
[12] F. Hertel et al. Extending the Tally-hiding Ordinos System: Implementations for Borda, Hare-Niemeyer, Condorcet, and Instant-Runoff Voting. *Cryptology ePrint Archive*, 2021.
[13] V. Cortier, P. Gaudry, and Q. Yang. A Toolbox for Verifiable Tally-hiding E-voting Systems. In *ESORICS*, 2022.
[14] N. Huber et al. Kryvos: Publicly Tally-Hiding Verifiable E-Voting. In *CCS*, 2022.
[15] J. Benaloh, T. Moran, L. Naish, K. Ramchen, and V. Teague. Shuffle-Sum: Coercion-Resistant Verifiable Tallying for STV Voting. *TIFS*, 4(4):685–698, 2009.
[16] B. Adida et al. Electing a University President using Open-audit Voting: Analysis of Real-world Use of Helios. *EVT/WOTE*, 2009.
[17] F. Hao et al. Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting. *JETS*, pages 1–25, 2014.
[18] S. Shahandashti and F. Hao. DRE-ip: a Verifiable E-voting Scheme without Tallying Authorities. In *ESORICS*, 2016.
[19] J. Lundell. Random Tie-breaking in STV. *Voting Matters*, 22:1–6, 2006.
[20] J. C. O'Neill. Tie-breaking with the Single Transferable Vote. *Voting matters*, 18 (14), 2004.
[21] H. M. Robert III, D. H. Honemann, T. J. Balch, D. E. Seabold, and S. Gerber. *Robert's Rules of Order Newly Revised.* PublicAffairs, 2020.
[22] F. Hao et al. End-to-End Verifiable E-voting Trial for Polling Station Voting. *IEEE S&P*, 18(6):6–13, 2020.
[23] J. Camenisch and M. Stadler. Proof Systems for General Statements about Discrete Logarithms. *Technical Report/ETH Zurich*, 260, 1997.
[24] J. Benaloh. Ballot Casting Assurance via Voter-initiated Poll Station Auditing. *EVT*, 2007.
[25] J. Benaloh. Verifiable Secret-ballot Elections. 1989.
[26] D. Bernhard et al. SoK: A Comprehensive Analysis of Game-based Ballot Privacy Definitions. In *IEEE S&P*, 2015.
[27] D.R.Stinson and M.Paterson. *Cryptography: Theory and Practice.* CRC press, 2018.
[28] K. Kurosawa and R. Nojima. Simple Adaptive Oblivious Transfer without Random Oracle. In *ASIACRYPT*, 2009.
[29] K James et al. Security Foundations for Application-Based Covert Communication Channels. In *IEEE S&P*, pages 1971–1986. IEEE, 2022.
[30] B. Schoenmakers and M. Veeningen. Universally Verifiable MPC from Threshold Homomorphic Cryptosystems. In *ACNS*, 2015.
[31] A. Fiat and Shamir A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Crypto*, 1986.

APPENDIX

A WELL-FORMEDNESS OF TALLYING PHASE

During the Tallying phase, we use Theorem 3.2 to verify that a matrix $\mathbf{V}_k^{(m)}$ is well-formed, i.e., it is the result of applying Algorithm 1 to $\mathbf{V}_k^{(m-1)}$ given a candidate $\alpha^{(m-1)}$ marked for elimination. Recall that the following must hold:

$$\bigvee_{l\in[1,n-m+2]}\left(\ \gamma\ \wedge\left(\bigwedge_{i\in[1,l-1]}\psi\right)\wedge\left(\bigwedge_{i\in[l,n-m+1]}\psi'\right)\ \right)$$

Where we have:

- $\gamma \leftarrow (v_{l\alpha^{(m-1)}})_k^{(m-1)} = 1 \wedge \sum_{j\neq\alpha^{(m-1)}}(v_{lj})_k^{(m-1)} = 0$
- $\psi \leftarrow (\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_i)_k^{(m-1)}$
- $\psi' \leftarrow (\mathbf{v}_i)_k^{(m)} = (\mathbf{v}_{i+1})_k^{(m-1)}$

Theorem 3.2 cannot be applied directly to the encrypted ballots $B_k^{(m)}$ and $B_K^{(m-1)}$. Instead, we transform γ, ψ and ψ' into zero knowledge statements Γ, Ψ and Ψ' as part of a proof of well-formedness P_{WF}^T. We define P_{WF}^T as below.

$$P_{WF}^T\left\{(B_{ij})_k^{(m)} = \left\langle (b_{ij})_k^{(m)}, (Y_{ij})_k^{(m)}\right\rangle\right\} =$$

$$P_K\Bigg\{(x_{ij})_k^{(m)}\ :$$

$$\left(\left(\log_{g_0}(b_{ij})_k^{(m)}/g_1 = \log_{g_1}(Y_{ij})_k^{(m)}\right)\right.$$

$$\left.\vee\left(\log_{g_0}(b_{ij})_k^{(m)} = \log_{g_1}(Y_{ij})_k^{(m)}\right)\right)$$

$$\wedge\bigvee_{l\in[1,n-m+2]}\Gamma\wedge\left(\bigwedge_{i\in[1,l-1]}\Psi\right)\wedge\left(\bigwedge_{i\in[l,n-m+1]}\Psi'\right)\Bigg\}$$

The first statement in P_{WF}^T ensures that $\mathbf{V}_k^{(m)}$ is a binary matrix. The final statement then ensures Theorem 3.2 holds between the two ballots $B_k^{(m)}$ and $B_k^{(m-1)}$.

We denote by Γ the following logical statement.

$$\log_{g_0}(b_{l\alpha^{(m-1)}})_k^{(m-1)}/g_1 = \log_{g_1}(Y_{l\alpha^{(m-1)}})_k^{(m-1)}$$
$$\wedge\log_{g_0}\prod_{j\neq\alpha^{(m-1)}}(b_{lj})_k^{(m-1)} = \log_{g_1}\prod_{j\neq\alpha^{(m-1)}}(Y_{lj})_k^{(m-1)} \qquad (\Gamma)$$

Γ utilises the additive property of the ElGamal ciphertexts to prove that a row l encodes the candidate $\alpha^{(m-1)}$ marked for elimination.

We then denote by Ψ the following logical statement.

$$\left(\log_{g_0}(b_{ij})_k^{(m)}/g_1 = \log_{g_1}(Y_{ij})_k^{(m)}\right.$$
$$\left.\wedge\log_{g_0}(b_{ij})_k^{(m-1)}/g_1 = \log_{g_1}(Y_{ij})_k^{(m-1)}\right)$$
$$\vee\left(\log_{g_0}(b_{ij})_k^{(m)} = \log_{g_1}(Y_{ij})_k^{(m)}\right.$$
$$\left.\wedge\log_{g_0}(b_{ij})_k^{(m-1)} = \log_{g_1}(Y_{ij})_k^{(m-1)}\right) \qquad (\Psi)$$

Ψ ensures that each entry in rows $i < l$ of $b_k^{(m)}$ and $b_k^{(m-1)}$ are the encryption either both 0 or both 1, i.e. the encryption of the same encoding of a candidate.

Finally, we denote by Ψ' the following logical statement.

$$\begin{aligned}
&\Big(\log_{g_0}(b_{ij})_k^{(m)}/g_1 = \log_{g_1}(Y_{ij})_k^{(m)} \\
&\wedge \log_{g_0}(b_{(i+1)j})_k^{(m-1)}/g_1 = \log_{g_1}(Y_{(i+1)j})_k^{(m-1)}\Big) \\
&\vee \Big(\log_{g_0}(b_{ij})_k^{(m)} = \log_{g_1}(Y_{ij})_k^{(m)} \\
&\quad \wedge \log_{g_0}(b_{(i+1)j})_k^{(m-1)} = \log_{g_1}(Y_{(i+1)j})_k^{(m-1)}\Big) \qquad (\Psi')
\end{aligned}$$

Ψ' ensures that each entry in all other rows $i > l$ of $b_k^{(m-1)}$ are the same encryption as those shifted upwards by one place in $b_k^{(m)}$.

P_{WF}^T should hold for all $i, j \in [1, n]$ and $m \in [1, n-1]$. All proofs of well-formedness are realised as proofs of knowledge P_K. These proofs are constructed using the standard techniques for creating conjunctive and disjunctive ZKPs [23]. We make each proof non-interactive by applying the Fiat-Shamir heuristic [31].