VERICONDOR: End-to-End Verifiable Condorcet 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

Hang Luo
Peking University
China
hang.luo@pku.edu.cn

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

ABSTRACT

Condorcet voting, first proposed by Marquis de Condorcet in the 18th century, chooses a winner of an election as one that defeats every other candidate by a simple majority. According to Condorcet's criterion, a Condorcet winner is the socially optimal choice in a multi-candidate election. However, despite the crucial importance of this voting system in social-choice theory, it has not been widely used in practical applications. This is partly due to the complex tallying procedure, and also the fact that several candidates may form a tie. Existing systems that provide online Condorcet voting services in the real world try to speed up the tallying process by collecting and tallying Condorcet ballots in a digital form. However, they require voters to completely trust the server. In this paper, we propose VERICONDOR, the first end-to-end verifiable Condorcet e-voting system without any tallying authorities. Our system allows a voter to fully verify the tallying integrity without involving any trustworthy tallying authorities and provides strong protection of the ballot secrecy. One main challenge in our work lies in proving the well-formedness of an encrypted ballot while being able to tally the ballots in a publicly verifiable yet privacy-preserving manner. We overcome this challenge by adopting a pairwise comparison matrix and applying a novel vector-sum technique to achieve exceptional efficiency. The overall computational cost per ballot is $O(n^2)$ where n is the number of candidates. This is probably the best that one may hope for given the use of a $n \times n$ matrix to record a Condorcet ballot. In case of a tie, we show how to apply known Condorcet methods to break the tie in a publicly verifiable manner. Finally, we present a prototype implementation and benchmark performance to show the feasibility of our system.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**.

KEYWORDS

E2E verifiability; Self-enforcing e-voting; Condorcet voting; Condorcet winner.

ACM Reference Format:
Luke Harrison, Samiran Bag, Hang Luo, and Feng Hao. 2022. VERICONDOR: End-to-End Verifiable Condorcet Voting without Tallying Authorities. In *Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security (ASIA CCS '22), May 30–June 3, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 13 pages.

1 INTRODUCTION

One of the most popular electoral systems used in the real world is the plurality voting, in which the candidate with the most votes wins the election. Whilst plurality voting has been widely adopted in practice, it is known to be prone to vote splitting. For example, Trump won none of his first 17 victories in the 2016 Republican primaries by a majority, but was still nominated because mainstream republicans cancelled each other by splitting the votes [1].

Condorcet voting was originally proposed by a French philosopher and mathematician, Marquis de Condorcet (1743 – 1794) as an improvement over plurality voting [2]. The use of a majority rule to decide a choice between two options was then well accepted. Condorcet was the first to extend this majority argument to the case of more than two options. Condorcet's criteria states that a candidate who defeats every other candidate by a simple majority is the *socially optimal* choice. Such a candidate is called a *Condorcet winner*. Conversely, a candidate who is defeated by every other candidate in a series of pairwise elections is called a *Condorcet loser*. Table 1 shows an example of ranking among Trump, Rubio and Kasich based on 2016 polls [1]. In this example, 40% of voters prefer Trump to Kasich, who in turn is preferred to Rubio. However, both Kasich and Rubio would defeat Trump in a head-to-head contest, as 60% of voters prefer either Kasich or Rubio to Trump. Hence, by the Condorcet criterion, Trump is a Condorcet loser, or in other words, a socially least optimal choice. However, he wins the plurality voting as the other two candidates have split the anti-Trump vote.

40%	**35%**	**25%**
Trump	Rubio	Kasich
Kasich	Kasich	Rubio
Rubio	Trump	Trump

Table 1: An example to illustrate vote splitting.

Condorcet voting is preferred when electing a socially optimal choice among multiple candidates is considered desirable. This voting method has been adopted by many groups and organizations, such as the Wikimedia Foundation, Debiant, Gentoo, Ubuntu, K Desktop Environment (KDE), the pirate party of Sweden, OpenStack, ICANN, ACM, IEEE, USENIX and Google for internal decisions and polls [3]. There is a free online Condorcet voting system, namely, the Condorcet Internet Voting Service (CIVS)[1] created and maintained by Andrew Myers of Cornell University. CIVS has been used for more 30,000 elections with more than 500,000 votes cast. OpaVote[2] is another online platform, providing Condorcet voting as a paid commercial service.

Although Condorcet voting is regarded as one of the most important electoral systems in social choice theory, several difficulties have prevented it from being used in wider applications. First, as compared to plurality voting, Condorcet voting is significantly more complex in counting. When ballots are cast on paper, manual counting can be a tedious and error-prone process. The use of digital technologies can speed up counting, but it introduces threats that digital data can be easily modified to alter the election result. Second, in some elections, a Condorcet winner may not exist, e.g., there may exist a circular relation that Candidate A is preferred over Candidate B, Candidate B is preferred over Candidate C and Candidate C is preferred over Candidate A. This circular relation forms a tie. A few solutions have been proposed to break the tie in the event that a Condorcet winner does not exist [3]. However, how these solutions break the tie in a publicly verifiable and privacy-preserving manner has not been investigated. This will be addressed in our work.

In this paper, we will investigate a fully verifiable Condorcet e-voting system, addressing both problems above. Our design builds on the well-established notion of *end-to-end verifiability* [4], fulfilling the following requirements.

(1) *Cast as intended* – any voter is able to verify that their votes do indeed represent their chosen candidate and not another;
(2) *Recorded as cast* – any voter is able to verify that their vote is recorded and included in the tallying process;
(3) *Tallied as recorded* – anyone (including any third-party observer) is able to verify that all the recorded votes are indeed tallied correctly.

Many E2E-verifiable voting systems have been proposed in the literature. Most of the proposed solutions require a set of tallying authority (TA), who are supposedly trustworthy individuals tasked to perform complex decryption and tallying operation. Examples of these solutions include Scantegrity [5], Scantegrity II [6], Prêt à voter [7], STAR-Vote [8], DEMOS [9], DEMOS-2 [10] and Helios [11]. However, in practice, choosing and managing the TAs has proved to be particularly difficult [12]. Recent advances in this field show that this problem can be overcome by building E2E verifiable voting systems that are free from any TAs. Researchers call this TA-free E2E voting paradigm "self-enforcing voting" (SEEV) [13]. Examples of the such systems include DRE-i [14], DRE-ip [15] and DRE-Borda [16]. Among them, DRE-i and DRE-ip are for plurality voting while DRE-Borda is for Borda-count voting.

While E2E verifiable systems for plurality voting have been extensively studied in the past, E2E verifiable solutions for Condorcet voting have received almost no attention. This is unsatisfactory given the crucial importance of Condorcet voting in electoral systems and social-choice theory. Existing online Condorcet voting systems (e.g., CIVS and OpaVote) that provide voting services to real-world voters are not E2E verifiable. In these systems, if the voting server is compromised, the tallying integrity of an election will be completely lost.

In this paper we propose VERICONDOR, the first E2E-verifiable Condorcet voting system without any tallying authorities. Our work is inspired by DRE-ip [15], which removes the need for TAs by cancelling out random factors introduced in the public key encryption process. We adopt a similar approach in our design. We note that DRE-ip is designed for plurality voting, but Condorcet voting is significantly more complex than plurality voting. A major challenge in our work lies in how to encrypt the ballot in a form that allows homomorphic tallying and meanwhile enables anyone to publicly verify the well-formedness of the encrypted ballot in a secure and efficient manner. We will explain how we have overcome this challenge by employing a novel vector-sum technique. Because of this technique, we are able to achieve exceptional efficiency. VERICONDOR will elect a Condorcet winner when one exists; in the event that a Condorcet winner does not exist, the system will elect an alternative winner based on a selection of Condorcet methods in a public verifiable and privacy-preserving manner.

We summarize our contributions as follows.

(1) We propose the first E2E verifiable Condorcet e-voting system without any TAs. Our system is also exceptionally efficient, incurring only $O(n^2)$ computation for each ballot where n is the number of candidates.
(2) We discuss how to elect an alternative winner, in the event that the Condorcet winner does not exist, based on a selection of Condorcet methods.
(3) We build an open-source prototype for the VERICONDOR system and present detailed experimental results to evaluate system performance.

The rest of the paper is organized as follows. In Section 2, we explain Condorcet voting in more detail as well as preliminaries required for our proposed system. Section 3 presents the VERICONDOR system, followed by security analysis and proofs in Section 4. In Section 5, we present a prototype of VERICONDOR and evaluate its system performance. Section 6 reviews the related work. Finally, Section 7 concludes the paper.

2 PRELIMINARIES

We now define how votes and tallies are represented in elections using Condorcet methods and introduce the cryptographic preliminaries required for our proposed VERICONDOR system.

2.1 Condorcet Voting

Elections using Condorcet methods typically require voters to rank the candidates in order of preference. For an n-candidate election with candidates belonging to the set $C = \{0, 1, ..., n-1\}$, each vote may be represented as a permutation $\mathbf{p} = (c_0, c_1, ..., c_{n-1})$ of the set C, where $c_a \in C$ for $a \in [0, n-1]$ and $c_a \neq c_b$ for all $a, b \in [0, n-1]$ and $a \neq b$. A candidate c_a is then *preferred* to a candidate c_b if $a < b$, i.e., candidate c_a appears before candidate c_b in $\mathbf{p}$ when $\mathbf{p}$ is read from left-to-right in descending order. We may illustrate this by considering a 3-candidate election with $C = \{0, 1, 2\}$. A voter may choose the permutation $(1, 2, 0)$ as their vote, meaning that they prefer candidate 1 to all other candidates, candidate 2 to candidate 0 and candidate 0 to no other candidates.

[1]https://civs1.civs.us/
[2]https://www.opavote.com/

With a ranked list of preferences, it is possible to construct an E2E verifiable voting system using one of the two trivial methods, but neither is desirable. The first is to simply encrypt the ranked list, pass the ciphertexts through a mix-net, and finally decrypt all ciphertexts after mixing, similar to an early version of Helios (1.0) [4]. Here TAs are required to run the mix-net servers, which is a non-trivial task. Furthermore, the decryption of all votes renders the system vulnerable to an Italian attack [4], in which a voter is coerced to choose an uncommon permutation of candidates so the coercer can verify it in the decrypted votes. (The mix-net approach in Helios 1.0 was replaced by homomorphicc tallying in Helios 2.0 [11].) The second method is to encode all of the $n!$ possible ranked lists as voter choices in the DRE-ip protocol [15]. But the $O(n!)$ complexity is clearly unscalable. Among $n!$ possible permutations, some of them will be considered uncommon or obscure. Therefore, concerns on an Italian attack still exist.

An alternative way to present a Condorcet vote is by using a *pairwise comparison matrix*, which is useful for simplifying the tallying process [17]. This matrix is also useful to minimize the information leakage, hence addressing concerns of an Italian attack. However, when the matrix is encrypted, how can we prove that an encrypted matrix is well-formed without leaking information about the plaintext vote? The solution to this problem is crucial for the E2E verifiability of the system. In this paper we refer to a matrix $\mathbf{V} = (v_{ij})$ as a pairwise comparison matrix if it satisfies the following properties:

$$\forall i \in C\colon v_{ii} = 0 \tag{P1}$$

$$\forall i, j \in C\colon v_{ij} = 0 \vee v_{ij} = 1 \tag{P2}$$

$$\forall i, j \in C,\ i \neq j\colon v_{ij} = 0 \Leftrightarrow v_{ji} = 1 \tag{P3}$$

(P1) simply states that no candidate can be preferred to themselves. (P2) states that each entry in the matrix $\mathbf{V}$ can only be 0 or 1. (P3) tells us that if a voter prefers candidate i to candidate j, then they do not prefer candidate j to candidate i and hence v_{ij} and v_{ji} cannot equal the same value.

Given a permutation $\mathbf{p}$ of candidates, we can construct a pairwise comparison matrix $\mathbf{V}$ by assigning $v_{ij} = 1$ and $v_{ji} = 0$ if candidate i is preferred to candidate j in $\mathbf{p}$. In the case that i and j represent the same candidate, i.e., $i = j$, then we assign $v_{ii} = 0$. Figure 1 illustrates this idea by showing the pairwise comparison matrix for the vote $(1, 2, 0)$.

$$(1,2,0) \quad \rightarrow \quad \begin{array}{c|c|c|c|} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 1 \\ \hline 2 & 1 & 0 & 0 \\ \hline \end{array} \quad \rightarrow \quad \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Figure 1: Obtaining a comparison matrix from $(1, 2, 0)$**.**

Assuming that there is a set of K voters in an election, we may represent the pairwise comparison matrix for a voter $k \in K$ by $\mathbf{V}_k$. Tallying the votes in an election is then straightforward and simply consists of computing $\sum_{k \in K} \mathbf{V}_k$ using matrix addition. We refer to the resulting matrix as the *sum matrix*. To illustrate, the sum matrix for the votes $(1, 2, 0)$, $(1, 0, 2)$ and $(0, 1, 2)$ is given in Figure 2.

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 1 & 0 & 0 \end{pmatrix}$$

Figure 2: An illustration of a sum matrix encompassing the three votes $(1, 2, 0)$**,** $(1, 0, 2)$**, and** $(0, 1, 2)$**.**

2.2 Cycles and Constraints

It may be the case that a Condorcet winner does not exist for an election at all; if an election determines that candidate 0 is preferred to candidate 1, candidate 1 is preferred to candidate 2 and candidate 2 is preferred to candidate 0 then there is no candidate who beats all other candidates. This is called a *Condorcet cycle* [2]. A Condorcet cycle essentially indicates a tie between certain candidates. To break the tie, various solutions have been proposed, including the Minimax method, the Schulze method, Copeland's method and Black's method. We will discuss how these methods may be integrated into our system in a publicly verifiable manner in Section 3.5.

When we represent a Condorcet vote in a comparison matrix, it is critical to ensure that the matrix, after being encrypted, is well-formed. Consider the following matrix as an example.

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{1}$$

The above matrix satisfies (P1), (P2) and (P3), but does not represent a valid ranked list. Here, candidate 0 is preferred to candidate 2, candidate 2 is preferred to candidate 1 and candidate 1 is preferred to candidate 0. This forms a cycle. Clearly, the vote is malformed.

We define a pairwise comparison matrix to be *valid* or *well-formed* if it represents one of the $n!$ fully ranked lists for an n-candidate election. An invalid pairwise comparison matrix is one that contains a cycle or breaks transitivity constraints. Transitivity constraints refer to the basic property that a voter's preferences are transitive, e.g., if $\mathbf{V}(0,1) = 1$ and $\mathbf{V}(1,2) = 1$ then we must have $\mathbf{V}(0,2) = 1$. Let us first consider the transitivity constraints for three candidates (out of a total of n candidates). We must take every triple of unique candidates (i, j, k), where $i, j, k \in C \wedge i \neq j \neq k$ and verify whether the following logical statements hold:

$$\begin{aligned} &\mathbf{V}(i,j) = 1 \wedge \mathbf{V}(j,k) = 1 \Rightarrow \mathbf{V}(i,k) = 1 \\ &\mathbf{V}(i,j) = 0 \wedge \mathbf{V}(j,k) = 0 \Rightarrow \mathbf{V}(i,k) = 0 \end{aligned} \tag{2}$$

We define each of the statements in Equation 2 as a *constraint*. To enforce each constraint will require a corresponding zero-knowledge proof (ZKP). We will also need a conjunctive ZKP to prove that all constraints are satisfied. To quantify the number of constraints, first let us consider the transitive relations among three candidates. It is not difficult to count that the number of constraints required is $\binom{n}{3} \times 3! \times 2 = 2n(n-1)(n-2)$. Similarly, for four candidates, the number of the constraints required to define the transitive relations is $\binom{n}{4} \times 4! \times 2$. In total, the number of constraints will be $\sum_{i=3}^{n} \binom{n}{i} \times i! \times 2$. With more candidates n, the size of the ZKP will grow exponentially. This is clearly unscalable.

We propose a much more efficient approach for verifying validity based upon an observation between valid pairwise comparison matrices and their corresponding row sums. We observe that the vectors of row sums for the valid matrix in Figure 1 and the invalid one in Equation 1 are $(0, 2, 1)$ and $(1, 1, 2)$ respectively; the former is a permutation of the set of candidates C, whilst the latter is not. This observation actually holds for any pairwise comparison matrix, which we will prove in Theorem 2.1.

Theorem 2.1. A pairwise comparison matrix $\mathbf{V}$ is valid if and only if $(\sum_{j=0}^{n-1} v_{ij})_{i=0}^{n-1}$, the vector of row sums for $\mathbf{V}$, is a permutation of the set of candidates C.

Proof ($\Rightarrow$). First, we prove that for a valid matrix, the vector of row sums is a permutation of C. Consider an n-candidate election with a set of candidates $C = \{0, ..., n-1\}$. Let $\mathbf{V} = (v_{ij})$ be a valid pairwise comparison matrix representing one of the possible $n!$ votes. We may write the vote (permutation) encoded by $\mathbf{V}$ as $\mathbf{p} = (c_0, c_1, \ldots, c_{n-1})$ where $c_a \in C$ for $a \in C$ and $c_a \neq c_b$ for all $a, b \in C$ and $a \neq b$. We consider each element in $\mathbf{p}$ one at a time, starting with c_0. Element c_0 is preferred to $c_1, c_2, \ldots, c_{n-1}$ by $\mathbf{p}$. Hence $v_{c_0c_1} = 1, v_{c_1c_0} = 0, \ldots, v_{c_0c_{n-1}} = 1, v_{c_{n-1}c_0} = 0$. There are $n-1$ entries of 1 in the row of $\mathbf{V}$ corresponding to c_0. Now consider c_1: c_1 is preferred to $c_2, \ldots, c_{n-1}$. Hence $v_{c_1c_2} = 1, v_{c_2c_1} = 0, \ldots, v_{c_1c_{n-1}} = 1, v_{c_{n-1}c_1} = 0$. There are exactly $n-2$ entries of 1 in the row of $\mathbf{V}$ corresponding to c_1. Repeating this process for each $c \in C$ results in each row of $\mathbf{V}$ having a unique number of entries of 1 and hence $\sum_{j=0}^{n-1} v_{ij}$ gives unique results for unique values of i, with results belonging to $[0, n-1]$. This is by definition a permutation over C.

($\Leftarrow$). Next, we show that if the vector of row sums in a matrix is a permutation of C, the matrix is valid. Suppose we have a pairwise comparison matrix $\mathbf{V}$ with the property that its vector of row sums is a permutation over a set of candidates $C = \{0, ..., n-1\}$. Denote the vector of row sums for $\mathbf{V}_k$ as $\mathbf{V}_k^{\Sigma}$. By definition each element in $\mathbf{V}^{\Sigma}$ must belong to C and $\mathbf{V}^{\Sigma}$ must contain no duplicated elements. Hence there is a unique maximum in $\mathbf{V}^{\Sigma}$, being the value $n-1$. Now consider each element in $\mathbf{V}^{\Sigma}$ in turn, starting with $n-1$: there must be an i_0 such that $\sum_{j=0}^{n-1} v_{i_0j} = n-1$. By the definition of a pairwise comparison matrix, row i_0 of $\mathbf{V}$ must consist of exactly $n-1$ entries of 1. Hence for all other rows i_k, where $0 \leq k < n-1$, the candidate represented by row i_0 is preferred to all other candidates represented by rows i_k. Now consider the next largest element of $\mathbf{V}^{\Sigma}$, being the unique value $n-2$: there must be an i_1 such that $\sum_{j=0}^{n-1} v_{i_1j} = n-2$. Row i_1 must consist of exactly $n-2$ entries of 1, again by definition of a pairwise comparison matrix as well as by our previous deduction that row i_0 consists of $n-1$ entries of 1, so entry $v_{i_1i_0}$ must be a 0. Hence the candidate represented by row i_1 is preferred to all other candidates i_k where $0 \leq k < n-2$. Repeating this process for each row results in the candidate represented by row i_0 being preferred to the candidate represented by row i_1, who is then preferred to the candidate represented by row i_2 and so on, down to the candidate represented by row i_{n-1}. We may write this as the permutation $\mathbf{p} = (i_0, i_1, \ldots, i_{n-1})$. This is one of the $n!$ possible votes for this election. This completes the proof. $\square$

Theorem 2.1 is a crucial result that provides an exceptionally efficient way to verify the validity of pairwise comparison matrices compared to the approaches that quantify transitivity constraints. Essentially, this theorem allows us to change a complex problem of proving the well-formedness of the comparison matrix to a simpler but equivalent problem of proving the the vector sum is a permutation of C. This will greatly simplify the zero-knowledge proofs (ZKPs) in VERICONDOR as we will explain below.

3 PROPOSED SOLUTION

We now provide an E2E-verifiable voting system for a Condorcet election without involving any TAs. We will describe the system in the context of voting at a polling station using a touch-screen direct recording electronic (DRE) machine. However, the underlying protocol can also be implemented for online voting.

3.1 Cryptographic Setting

Let p and q be two large primes such that $q \mid p-1$, i.e., q divides $p-1$. We define $\mathbb{G}_q$ to be the subgroup of prime order q of the group $\mathbb{Z}_p^*$. All the modular operations are performed with reference to the modulus p unless otherwise specified. Let g_0 and g_1 be two random generators of $\mathbb{G}_q$, whose discrete logarithm is unknown. We can first fix the first generator g_0 to be any non-identity element in $\mathbb{G}_q$ and compute the second generator g_1 based on a one-way hash function including g_0 and public contextual information (e.g., election title, date and candidates) in the input [18].

We assume the decision Diffie-Hellman (DDH) assumption to be hard in $\mathbb{G}_q$. We state the DDH assumption [19] as follows:

Assumption 3.1. 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 $\Omega = R$.

3.2 Requirements

Much like any other E2E voting system [4], we require a publicly accessible bulletin board (BB) to facilitate both vote auditing and E2E-verifiability. The DRE-machines are assumed to only have append-only access to the BB over a secure channel. We also assume that each DRE-machine is connected to a printer for printing paper-based receipts of voting for each voter. A mechanism for voters to rank candidates in order of preference is also required for each DRE-machine, such as by using a touch screen to allow voters to reorganise the order of candidates to their liking. In our current system, we only support full ranking of all candidates, and will leave support for partial ranking to future work.

3.3 Condorcet E-Voting

On the election day, a user is first authenticated at the polling station and then obtains an anonymous credential, such as a one-time passcode or a random smart card. The voter enters a private voting booth and logs onto the DRE machine with the obtained credential to start voting.

In VERICONDOR, casting a vote is done in two steps. In the first step, a voter ranks n candidates in the order of preferences and clicks the “next” button. The printer will print the first half of the receipt, containing the encrypted ballot. In the second step, the voter is prompted to choose “confirm” or “cancel” for the selection. In case of “confirm”, the printer will print the second half of the receipt, containing a confirmation message that the vote has been cast. In case of “cancel”, the printer will print the second half of the receipt containing the selected order of candidates in plaintext and random factors used in the ballot encryption to allow voter-initiated auditing [20]. All receipts are digitally signed to prove data authenticity and are published on the bulletin board. We will explain each of the two steps in more detail below.

In the first step, after the voter has ranked all n candidates, the system will represent the ranked list using an $n \times n$ pairwise comparison matrix. Assuming that there is a set of K voters in an election, we denote the plaintext pairwise comparison matrix for a voter $k \in K$ as $\mathbf{V}_k = (v_{ij})_k$. The DRE-machine holds two $n \times n$ matrices $\mathbf{S} = (s_{ij})$ and $\mathbf{T} = (t_{ij})$ which are both initialised to $\mathbf{0}_{n,n}$. Once a voter clicks the "next" button, the DRE-machine generates an $n \times n$ matrix $\mathbf{X}_k = (x_{ij})_k$ containing random numbers as follows:

$$\forall i \in C \colon x_{ii} = 0 \tag{X1}$$

$$\forall i, j \in C \colon i \neq j \Rightarrow x_{ij} \in_R \mathbb{Z}_q^* \tag{X2}$$

(X1) states that the main diagonal of $\mathbf{V}_k$ always consists of 0s. (X2) represents a selection of values taken uniformly at random from $\mathbb{Z}_q^*$. The DRE translates the ranked list into an $n \times n$ matrix and produces an encrypted ballot $B_k = \langle b_k(i,j), Y_k(i,j) \rangle_k$ where:

$$\forall i \in C \colon b_k(i,i) = 1 \tag{BC1}$$

$$\forall i, j \in C \colon i \neq j \Rightarrow b_k(i,j) = g_0^{x_{ij}} g_0^{v_{ij}} \tag{BC2}$$

$$\forall i, j \in C \colon Y_k(i,j) = g_1^{x_{ij}} \tag{BC3}$$

In these equations, "BC" refers to *ballot construction.* (BC1) states that the encryption of a ranking between the same candidate is just 1. This is because for any candidate i, both x_{ii} and v_{ii} are 0 and hence $b_k(i,i)$ must equal 1. (BC2) and (BC3) represent the main encryption of $\mathbf{V}_k$ using $\mathbf{X}_k$, g_0 and g_1, fulfilling the following logical relation (which can be enforced by a disjunctive ZKP [21]):

$$\left(\log_{g_0} b_k(i,j)/g_0 = \log_{g_1} Y_k(i,j)\right) \vee \left(\log_{g_0} b_k(i,j) = \log_{g_1} Y_k(i,j)\right)$$

The DRE-machine additionally constructs a set of non-interactive ZKPs (NIZKPs) confirming the well-formedness of the ballot B_k; we discuss the details in Section 3.4. Each ballot B_k and its corresponding set of NIZKPs are printed on a paper receipt, along with a digital signature to prove authenticity. (In a practical implementation, it is possible to print only a hash rather than the full data on the user receipt, and publish the hash together with the full cryptographic data on BB for public verification [18].)

In the second step, the voter can choose to either audit (i.e., cancel) or confirm their ballot. In the case of auditing the ballot, the DRE-machine additionally prints the voter's ranking of candidates, and the random matrix $\mathbf{X}_k$ on the same receipt along with a digital signature. The ballot B_k is included in a set $\mathbb{A}$ of audited ballots and the entire content of the receipt is posted to the BB as an audited ballot. The voter can check that the printed ranking of candidates reflects their original selection in Step 1; if not, a dispute should be raised immediately to the election staff in the polling station. The voter does not need to understand any cryptographic data printed on the receipt. They just need to check the same receipt is published on the bulletin board.

In the case that the voter chooses to confirm the ballot, the DRE-machine updates the matrices $\mathbf{S}$ and $\mathbf{T}$ as follows:

$$\forall i, j \in C \colon s_{ij} = s_{ij} + x_{ij} \tag{T1}$$

$$\forall i, j \in C \colon t_{ij} = t_{ij} + v_{ij} \tag{T2}$$

The DRE-machine then securely deletes $\mathbf{X}_k$ and $\mathbf{V}_k$. The machine will print a message to the voter receipt to confirm that the ballot has been cast and will publish the receipt on the BB. The voter just needs to check that the same receipt is published. The ballot B_k will be included in a set $\mathbb{C}$ of confirmed ballots recorded on the BB.

When the election day finishes, the DRE-machine publishes $\mathbf{S}$ and $\mathbf{T}$ on the BB. Here $\mathbf{T}$ is the aggregated result for the comparison matrix. To verify the tallying integrity of this result, anyone can perform the following checks: 1) the NIZKPs of well-formedness for each ballot hold; 2) the digital signatures are valid; 3) and the following equations hold:

$$\forall i, j \in C \colon g_0^{s_{ij}} g_0^{t_{ij}} = \prod_{k \in \mathbb{C}} b_k(i,j) \tag{TV1}$$

$$\forall i, j \in C \colon g_1^{s_{ij}} = \prod_{k \in \mathbb{C}} Y_k(i,j) \tag{TV2}$$

We use "TV" to refer to *tally verification.* (TV1) and (TV2) can be performed by anyone with read access to the public BB.

3.4 NIZKPs of Well-formedness

NIZKPs are necessary to ensure the well-formedness of ballots. We define a ballot $B_k = \langle b_k(i,j), Y_k(i,j) \rangle$ to be *well-formed* if $b_k(i,j)$ is the encryption of a pairwise comparison matrix and this pairwise comparison matrix is valid: representing one of the $n!$ possibilities for an n-candidate election. We will apply Theorem 2.1 to verify if the pairwise comparison matrix $\mathbf{V}_k$ is valid by computing its vector of row sums $\mathbf{V}_k^{\Sigma}$ and verifying whether this vector is a permutation over the set of candidates C. Based on Bag et al. [16], it is sufficient to prove the following logical statement:

$$\bigwedge_{c \in C} c \in \mathbf{V}_k^{\Sigma}$$

These relations prove that each $c \in C$ is an element of the vector $\mathbf{V}_k^{\Sigma}$. Given that C consists of only distinct values, and the cardinality of C is equal to the length of $\mathbf{V}_k^{\Sigma}$, it follows that C must be a permutation of $\mathbf{V}_k^{\Sigma}$ and equivalently that $\mathbf{V}_k^{\Sigma}$ is a permutation of C. These relations are equivalent to the following logical statement:

$$\bigvee_{i \in C} b_k^{\Pi}(i) = g_0^{\mathbf{X}_k^{\Sigma}(i) + j}$$

Where $b_k^{\Pi}(i) = \prod_{j=0}^{n-1} b_k(i,j)$ and $\mathbf{X}_k^{\Sigma}(i) = \sum_{j=0}^{n-1} \mathbf{X}_k(i,j)$ for some $i \in C$. We combine the above logical statement with four other logical statements to prove that a plaintext pairwise comparison matrix b_k also satisfies (P1), (P2) and (P3) when encrypted; this produces the proof of well-formedness $P_{WF}\{B_k = \langle b_k, Y_k \rangle\}$.

$$P_{WF}\{B_k = \langle b_k, Y_k \rangle\} =$$

$$P_K\Big\{\mathbf{X}_k \colon Y_k(i,j) = g_1^{x_{ij}} \tag{C1}$$

$$\wedge\ b_k(i,i) = g_0^{x_{ii}} \tag{C2}$$

$$\wedge\ \left(b_k(i,j) = g_0^{x_{ij}} \vee b_k(i,j)/g_0 = g_0^{x_{ij}}\right) \tag{C3}$$

$$\begin{aligned} \wedge\ \big(i \neq j \Rightarrow (&b_k(i,j) = g_0^{x_{ij}} \wedge b_k(j,i)/g_0 = g_0^{x_{ji}}) \\ &\vee (b_k(i,j)/g_0 = g_0^{x_{ij}} \wedge b_k(j,i) = g_0^{x_{ji}})\big) \end{aligned} \tag{C4}$$

$$\wedge\ \bigvee_{i \in C} b_k^{\Pi}(i) = g_0^{\mathbf{X}_k^{\Sigma}(i)+j}\Big\} \tag{C5}$$

For brevity we refer to this entire proof as just $P_{WF}\{B_k\}$ from now on. $P_{WF}\{B_k\}$ is realised as a proof of knowledge $P_K\{\mathbf{X}_k\}$ consisting of five conjunctive statements labelled (C1) through to (C5) above. We construct the full proof of knowledge by starting with proofs based upon Schnorr's signature scheme [22]. These proofs are then combined using standard techniques to create proofs of conjunctive knowledge and disjunctive knowledge [21, 23]. We also adopt the ZKP proposed by Bag et al. [16] for proving that one set is a permutation of another. Bag et al's technique is simple and reasonably efficient with a $O(n^2)$ complexity. In future work, we will investigate potentially improving Bag et al.'s scheme to $O(n \log n)$ by using BulletProof [24], but this will make no difference to the overall $O(n^2)$ complexity in our system, which is determined by the encryption of the matrix. These ZKPs are then made non-interactive by applying the Fiat-Shamir heuristic [25].

3.5 Electing a Winner

In a Condorcet election, several candidates may end up with a tie (forming a Condorcet circle). To break the tie, several Condorcet methods have been proposed in the past literature, such as Black's method [26], the Minimax method [27, 28], the Schulze method [3], Copeland's method [29], Ranked Pairs [30], Dodgson's method [31] and the Kemeny-Young method [32, 33]. We now discuss how our proposed VERICONDOR system could be used in conjunction with existing Condorcet methods, to elect an alternative winner in a publicly verifiable manner, in the event that there is no Condorcet winner for an election. In particular we focus upon Black's method, the Minimax method, the Schulze method, Copeland's method and Ranked Pairs. We do not consider either Dodgson's method or the Kemeny-Young method as these two methods both require solving an NP-hard problem when determining the Condorcet winner [34] and hence are much less efficient than the other listed methods. The methods which we do consider each provide different degrees of simplicity as well as satisfying different voting system criteria and hence anyone running an election using VERICONDOR can choose a suitable and efficient Condorcet method for their election.

Black's method. Black's method uses the Borda count system in the event that there is no Condorcet winner for an election [26]. Adapting VERICONDOR to support Black's method is straightforward; we simply run VERICONDOR and DRE-Borda [16] in parallel. Running these two methods in parallel is necessary as Borda count requires additional information concerning a number of points given to each candidate; this information cannot be feasibly acquired from the comparison matrix. Running VERICONDOR in parallel with DRE-Borda will increase the computational cost since two different electoral methods need to be run simultaneously.

The Minimax method. The Minimax method elects the winner whose greatest pairwise defeat is smaller than the greatest pairwise defeat of any other candidate [27, 28]. Electing a winner using the Minimax method in VERICONDOR is straightforward since the aggregated pairwise comparison matrix contains all information needed; the winner is the result of $\arg\min_{i \in C}(\max_{j \in C}(\mathbf{T}(j, i)))$. The benefits of using the Minimax method are its simplicity as well as its efficiency; electing a winner using the Minimax method has a runtime of $O(n^2)$.

The Schulze method. The Schulze method may be divided into two stages [3]; the first stage determines potential winners and the second stage computes a Tie-Breaking Ranking of Candidates (TBRC) to elect a winner if there are multiple potential winners.

It is straightforward to apply the first stage of the Schulze method to VERICONDOR. We first define a *path* from a candidate i to a candidate j as a permutation $\mathbf{p}$ where $\mathbf{p}(0) = i$ and $\mathbf{p}(n-1) = j$. We then define the *strength* of a path $\mathbf{p}$ as $s(\mathbf{p}) = \min_{l \in C}(\mathbf{T}(\mathbf{p}(l), \mathbf{p}(l+1)) - \mathbf{T}(\mathbf{p}(l+1), \mathbf{p}(l)))$. Denote by P_{ij} the set of all paths between two candidates i and j. The output of the first stage is then a matrix $\mathbf{W} = (w_{ij})$ where each w_{ij} is the strength of the strongest path from candidate i to candidate j, i.e., $w_{ij} = \max_{\mathbf{p} \in P_{ij}}(s(\mathbf{p}))$. We say that candidate i is a *potential winner* if and only if $w_{ij} \geq w_{ji}$ for every other candidate j. The strongest paths, and hence the potential winners, may be calculated using the Floyd-Warshall algorithm [3]. The Floyd-Warshall algorithm is an efficient algorithm with a runtime of $O(n^3)$ to compute all strongest paths.

There are a couple of different approaches which may be used to break ties as part of the second stage of the Schulze method. Two candidates i and j may be declared indifferent in a ranking using the Schulze method if the weakest link in the strongest path from i to j is the same link as the weakest link in the strongest path from j to i [3]. In this case we may declare the weakest link as *forbidden* and recalculate the strongest paths, avoiding any forbidden links. This is repeated until no forbidden links are used as part of any strongest paths. This approach is straightforward to apply to the final tally matrix $\mathbf{T}$ of VERICONDOR and only requires an extension to the first stage to account for recomputation of strongest paths containing forbidden links. This approach however only computes a partial order of candidates [3]; Schulze proposes an additional tie-breaking approach that may be used to compute a total order of candidates. The alternative approach requires selection of votes at random and their rankings used to create a Tie-Breaking Ranking of Links (TBRL) before the TBRC is computed [3]. This is not compatible with VERICONDOR as the DRE-machine securely deletes each individuate vote $\mathbf{V}_k$ after the vote is confirmed and hence this information is not available for computing a total order of candidates. Only a partial order of candidates is possible using the Schulze method in conjunction with VERICONDOR.

Copeland's method. Copeland's method assigns a number of points to a candidate depending upon their number of pairwise victories, pairwise ties and pairwise defeats [35]. We may model this precisely using a results matrix; in a pairwise comparison between a candidate i and a candidate j, let $\Gamma = (\gamma_{ij})$ and define γ_{ij} as below.

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \mathbf{T}(i, j) > \mathbf{T}(j, i) \\ \frac{1}{2} & \text{if } \mathbf{T}(i, j) = \mathbf{T}(j, i) \\ 0 & \text{if } \mathbf{T}(i, j) < \mathbf{T}(j, i) \end{cases}$$

The Copeland score for a candidate i is computed as $\sum_{j \in C} \gamma_{ij}$ and the candidate with the highest Copeland score wins the election. In the event that the Copeland score for a candidate is $n-1$, then this candidate is also the Condorcet winner. Copeland's method is simple to utilize as part of VERICONDOR. It is also a flexible method; the number of points assigned as part of the definition of γ_{ij} may be changed for convenience [36]. For example, tallying could be simplified to make use of only integer additions by assigning points from $\{1, 0, -1\}$ or $\{2, 1, 0\}$ as opposed to assigning points from $\{1, \frac{1}{2}, 0\}$. The disadvantage with using Copeland's method is that it has no associated tie-breaking procedure and hence one must be decided upon in the event that Copeland's method produces multiple winners. Borda count could be used to break ties resulting from Copeland's method, however, like Black's method, this requires running VERICONDOR and DRE-Borda in parallel.

Ranked Pairs. The Ranked Pairs method begins by sorting pairs of candidates; a pair of candidates (i_0, j_0) is ranked higher than a pair of candidates (i_1, j_1) if $\mathbf{T}(i_0, j_0) > \mathbf{T}(i_1, j_1)$ [30]. In the event where $\mathbf{T}(i_0, j_0) = \mathbf{T}(i_1, j_1)$, a method is needed to break the tie. This may be done by firstly computing a TBRC and using the TBRC to create a Tie-Breaking Ranking of Pairs (TBRP) [37]. If $\mathbf{T}(i_0, j_0) = \mathbf{T}(i_1, j_1)$ and $i_0 \neq i_1$, then the TBRP ranks (i_0, j_0) higher if i_0 is ranked higher in the TBRC than i_1. If $i_0 = i_1$, then the TBRP ranks (i_0, j_0) higher if j_0 is ranked higher in the TBRC than j_1. Tideman proposes computing the TBRC by selecting a voter at random and using their vote to break the tie [30]; this is not possible to perform within VERICONDOR due to each $\mathbf{V}_k$ being securely deleted by the DRE-machine. The TBRC may be computed by randomly generating a permutation of candidates, however this would reward nomination of *clones* [30]: candidates who are similar to existing candidates and whose addition to the election may result in a different winner for the election.

Once all pairs of candidates are sorted, a final ranking of candidates may then be constructed. The candidate who beats the other candidate in the first pair ranked highest in the sorted list is added to the final ranking first. Then the pair ranked next highest is considered and its winning candidate added to the final ranking provided that this does not cause a cycle [30]; this may be performed by representing the final ranking as a directed graph with candidates as nodes and edges as pairs of candidates and checking for a cycle using Depth First Search (DFS). The final ranking is complete once all pairs have been considered and the overall winner of an election using Ranked Pairs is the candidate at the beginning of the final ranking; the winner may be determined using a topological sort on the directed (acyclic) graph representing the final ranking. The use of DFS and topological sort to compute the final ranking means that Ranked Pairs is an efficient method to pair with VERICONDOR as both DFS and topological sort have a runtime of $O(n^2)$.

Table 2 summarizes the support for five different Condorcet methods to elect an alternative winner in VERICONDOR in the event that a Condorcet winner does not exist. The pros and cons of each method have been well studied in the past. Here, we mainly focus on whether these methods can be conducted in a publicly verifiable yet privacy-preserving manner. Our analysis shows that the tallying result in the pairwise comparison matrix is sufficient to break a tie in the general case, although for Black's and Copeland's methods, a DRE-Borda count system needs to be run in parallel, which increases computation. For Schulze's method and Ranked Pairs, under certain conditions, they need the access to the original individual ballots to decide an alternative winner, but that is not possible in VERICONDOR since VERICONDOR only stores the aggregated results in the comparison matrix, not individual votes.

Method	Partial	Total
Black		✓
Minimax		✓
Schulze	✓	
Copeland		✓
Ranked Pairs	✓	

Table 2: Summary of support for different Condorcet methods in DRE-Condorcet.

4 SECURITY ANALYSIS

In this section we prove the correctness and E2E-verifiability of VERICONDOR. We also prove that our system is secure against an adversary who attempts to learn the plaintext values of individual honest voters via collusion with η dishonest voters. In particular we prove that the adversary with η colluding voters can only learn the partial tally of the $n - \eta$ honest votes based on the DDH assumption. Since we require a secure hash function for transforming an interactive ZKP to a non-interactive ZKP based Fiat-Shamir heuristics, our proofs are in a random oracle model.

4.1 E2E-Verifiability

We show VERICONDOR satisfies the three requirements in end-to-end verifiability. It is straightforward to see that our system satisfies the "cast as intended" requirement based on the well-established voter-initiated auditing technique [20]. The DRE-machine commits to the encrypted ballot by printing it on the paper receipt. In the case of auditing (i.e., when the voter chooses to cancel the section), the DRE-machine reveals the randomness $\mathbf{X}_k$ and the ranked list in plaintext on the same receipt, enabling the voter to verify that the ranked list truthfully reflects the intended vote. Since the receipt is also published on the bulletin board, anyone will be able to verify that the initially committed ciphertext is a truthful encryption of the ranked list based on the revealed randomness $\mathbf{X}_k$. Any voter may choose to audit their vote for any number of times, which cannot be predicted by the DRE-machine.

In the case of confirming a vote, the voter obtains a receipt for the confirmed vote and can check that the same receipt is published on the BB. This fulfills the "recorded as cast" requirement.

Finally, we show VERICONDOR satisfies the "tallied as recorded" requirement. Theorem 4.1 shows that if the ballots are well-formed, i.e., $P_{WF}\{B_k\}$ holds, and the tally verification equations (TV1) and (TV2) hold, then anyone is able to verify that the final tally on the BB is the correct tally representing the matrix addition of all confirmed votes, i.e., every $\mathbf{V}_k$ for $k \in \mathbb{C}$.

Theorem 4.1. In VERICONDOR, assuming that all proofs of well-formedness are valid, if $\forall k \in K\colon P_{WF}\{B_k\}$ holds and additionally $\forall i, j \in C\colon \prod_{k\in\mathbb{C}} b_k(i,j) = g_0^{s_{ij}} g_0^{t_{ij}} \wedge \prod_{k\in\mathbb{C}} Y_k(i,j) = g_1^{s_{ij}}$ also holds, then the reported tally $\mathbf{T}$ is the correct tally of all confirmed ballots in $\mathbb{C}$ on the BB.

Proof. Suppose that $\forall k \in K\colon P_{WF}\{B_k\}$ holds and also that $\forall i, j \in C\colon \prod_{k\in\mathbb{C}} Y_k(i,j) = g_1^{s_{ij}}$ holds. We show that $\prod_{k\in\mathbb{C}} b_k(i,j) = g_0^{s_{ij}} g_0^{t_{ij}}$ holds if and only if $t_{ij} = \sum_{k\in\mathbb{C}} v_{ij}$ also holds for all $i, j \in C$.

(⇒) Suppose $\prod_{k\in\mathbb{C}} b_k(i,j) = g_0^{s_{ij}} g_0^{t_{ij}}$ for all $i,j \in C$. By definition of $b_k(i,j)$, we have $b_k(i,j) = g_0^{x_{ij}} g_0^{v_{ij}}$ and hence $\prod_{k\in\mathbb{C}} b_k(i,j) = \prod_{k\in\mathbb{C}} g_0^{x_{ij}} g_0^{v_{ij}} = g_0^{\sum_{k\in\mathbb{C}} x_{ij}} g_0^{\sum_{k\in\mathbb{C}} v_{ij}}$. By applying (TV2), we have $s_{ij} = \sum_{k\in\mathbb{C}} x_{ij}$. It is then clear that $t_{ij} = \sum_{k\in\mathbb{C}} v_{ij}$.

(⇐) Suppose $t_{ij} = \sum_{k\in\mathbb{C}} v_{ij}$ for all $i,j \in C$. By definition of $b_k(i,j)$, we have $b_k(i,j) = g_0^{x_{ij}} g_0^{v_{ij}}$ and also $\prod_{k\in\mathbb{C}} b_k(i,j) = g_0^{\sum_{k\in\mathbb{C}} x_{ij}} g_0^{\sum_{k\in\mathbb{C}} v_{ij}}$. By applying (TV1) and our initial assumption we get the result $\prod_{k\in\mathbb{C}} b_k(i,j) = g_0^{s_{ij}} g_0^{t_{ij}}$. This completes the proof. □

4.2 Ballot Secrecy

We now consider the notion of *ballot secrecy* for VERICONDOR, which describes the natural requirement of a voting system in preserving the privacy of votes cast in an election. We make use of Benaloh's definition of privacy [38] and define privacy to be maintained if a set of η colluding voters has a negligible chance to distinguish between any two elections where both elections have the same partial tally of votes.

Assumption 4.1. Let us consider the following security experiment $Exp_{\mathcal{A}}^{RND}(\lambda)$. For any two elements $g^a, g^b \in \mathbb{G}_q$, let us define $DH_g(g^a, g^b) = g^{ab}$.

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

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

In the experiment, the challenger first randomly selects two elements g, and A. from $\mathbb{G}_q$. It then invokes the adversary $\mathcal{A}$ with these elements. $\mathcal{A}$ is given oracle access to O. O may be queried $poly(\lambda)$ times. On every query to O, it selects a random B from $\mathbb{G}_q$, and computes $DH_g(A, B)$. It then returns B, and either $DH_g(A, B)$ or a random element from $\mathbb{G}_q$ depending upon a secret bit d chosen by the challenger.

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

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

For any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{RND}(\lambda) \le negl(\lambda)$.

Lemma 4.1. The DDH assumption implies assumption 4.1.

Proof. This Lemma is proved as Lemma 4 in [39]. □

Assumption 4.2. Let us consider the following security experiment $Exp_{\mathcal{A}}^{RND1}(\lambda)$. In this experiment, the adversary passes two inputs to the oracle O, whenever it is called. Each of the two inputs is a bit. The oracle O randomly selects one of them and computes Ω_d as shown in the description. Everything else is same as the experiment $Exp_{\mathcal{A}}^{RND}(\lambda)$.

```
$Exp_{\mathcal{A}}^{RND1}(\lambda)$
$g \xleftarrow{\$} \mathbb{G}_q$
$A \xleftarrow{\$} \mathbb{G}_q$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(g, A)$
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 $(B, \Omega_d)$
```

The advantage of an adversary $\mathcal{A}$, against $Exp_{\mathcal{A}}^{RND1}(\lambda)$ is then given as

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

For any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{RND1}(\lambda) \le negl(\lambda)$.

Lemma 4.2. Assumption 4.1 implies 4.2.

Proof. The lemma can be easily proven by application of the triangle inequality for computational indistinguishability [40], that is to say for any three distributions D_0, D_1, D_2, the inequality $|D_0 - D_2| \le |D_0 - D_1| + |D_1 - D_2|$ holds. Using the triangle inequality, one can show that $Adv_{\mathcal{A}}^{RND1}(\lambda) \le 2 * Adv_{\mathcal{A}}^{RND}(\lambda)$. □

Definition 4.1. Consider the security experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$. In this experiment, the challenger first chooses two random generators g, and $\tilde{g}$. It then creates two bulletin boards BB_0, and BB_1, both are initialized to empty. It also stores three variables X, V_0, and V_1 that are initialized to 0. The challenger then invokes $\mathcal{A}_0$. $\mathcal{A}_0$ is given access to the oracle O. $\mathcal{A}$ passes two inputs $v_0 \in \{0,1\}$, and $v_1 \in \{0,1\}$ to the oracle O, every time it is called. O selects random $x \in \mathbb{Z}_p$, and generates c_0, and c_1 as shown in the experiment. It stores c_0, and c_1 in BB_0, and BB_1. It stores the cumulative values of the selected randomnesses in X. Similarly it stores the cumulative values of v_0, and v_1 in V_0, and V_1 respectively. When $\mathcal{A}_0$ returns, the challenger invokes $\mathcal{A}_1$ with X, and one of BB_0, and BB_1. The goal of $\mathcal{A}_1$ is to identify the correct bulletin board. $\mathcal{A}$ wins the game if $\mathcal{A}_1$ can identify the bulletin board, and if $V_0 = V_1$.

```
$Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$
$\tilde{g}, g \xleftarrow{\$} \mathbb{G}_q$
$BB_0 = BB_1 = \emptyset$
$V_0 = V_1 = 0$
$X = 0$
$st \leftarrow \mathcal{A}_0^{O(\cdot,\cdot)}(\tilde{g}, g)$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}_1(st, BB_d, X)$
Return $(V_0 = V_1) \wedge (d = d')$
```

```
$O(v_0, v_1)$
$x \xleftarrow{\$} \mathbb{Z}_p$
$c_0 \leftarrow (g^x g^{v_0}, \tilde{g}^x)$
$c_1 \leftarrow (g^x g^{v_1}, \tilde{g}^x)$
$X \leftarrow X + x$
$BB_i \leftarrow BB_i \bigcup \{c_i\} : i = 0,1$
$V_i \leftarrow V_i + v_i : i = 0,1$
```

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

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

Lemma 4.3. For any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have $Adv_{\mathcal{A}}^{IND-Vote}(\lambda) \le negl(\lambda)$.

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B}$, against the security experiment $Exp_{\mathcal{A}}^{RND1}(\lambda)$ of Assumption 4.2. $\mathcal{B}$ works as follows; it receives as input g, and A. It invokes $\mathcal{A}$ with (g, A). Whenever $\mathcal{A}_0$ makes a query to the oracle O with input v_0, and v_1, $\mathcal{B}$ also makes a similar query in the experiment $Exp_{\mathcal{B}}^{RND1}(\lambda)$. $\mathcal{B}$ receives (B, Ω_d) as the value returned by the oracle in $Exp_{\mathcal{B}}^{RND1}(\lambda)$. We denote by (B_i, Ω_{di}), the returned value against the ith oracle query. Let us suppose that $\mathcal{A}_0$ makes a maximum of n queries in $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$. Then $n \in poly(\lambda)$ must hold. $\mathcal{B}$ randomly selects an $\eta \in [2, n]$. When $\mathcal{A}_0$ makes the ηth query, $\mathcal{B}$ computes $B_\eta = A^X / \prod_{i=1}^{\eta-1} B_i$, and $\Omega_{d\eta} = g^{X+V} / \prod_{i=1}^{\eta-1} \Omega_{di}$. Here, $V = \sum_{i=1}^{\eta} v_{0i} = \sum_{i=1}^{\eta} v_{1i}$, and $X \xleftarrow{\$} \mathbb{Z}_p$. If η is the last oracle query of $\mathcal{A}_0$, then $\mathcal{B}$ generates $BB_d = \{(\Omega_{di}, B_i) : i \in [1, \eta]\}$. If η is not the last oracle query of $\mathcal{A}_0$, then $\mathcal{B}$ aborts and returns a random bit. Else $\mathcal{B}$ invokes $\mathcal{A}_1$ with BB_d. If $\mathcal{A}_1$ can identify d, $\mathcal{B}$ can win the game.

We now calculate the winning probability of $\mathcal{B}$. $\mathcal{B}$ can win the game if $\mathcal{A}_1$ makes a total of η oracle queries, and $\mathcal{A}_1$ can identify d. If η is not the total number of oracle queries of $\mathcal{A}_0$, then the probability that $\mathcal{B}$ wins the game is $\frac{1}{2}$. Again, the probability that η will be the last oracle query of $\mathcal{A}_0$ will be $1/poly(\lambda)$, since η is chosen randomly from $[2, n]$. Thus, we may write,

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

That is to say,

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

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

Lemma 4.4. *Given $g_0, g_1 \in \mathbb{G}_q$ and $\mathbf{X}'_k = (x'_{ij})$ where $x'_{ij} = g_1^{x_{ij}}$ for all $i, j \in C$, the two BBs BB_0 and BB_1, given in Tables 3 and 4 respectively, are indistinguishable where:*

(1) $\forall \kappa \in [0, \eta-1] \wedge \forall i, j \in C\colon \mathbf{V}_\kappa(i, j) \in \{0, 1\} \wedge \mathbf{V}'_\kappa(i, j) \in \{0, 1\}$,
(2) $\forall \kappa \in [0, \eta - 1]\colon$ each of $\mathbf{V}_\kappa$ and $\mathbf{V}'_\kappa$ are valid pairwise comparison matrices,
(3) $\sum_{\kappa=0}^{\eta-1} \mathbf{V}_\kappa = \sum_{\kappa=0}^{\eta-1} \mathbf{V}'_\kappa$,
(4) $\forall i, j \in C\colon s_{ij} = \sum_{\kappa=0}^{\eta-1} \mathbf{X}_\kappa(i, j)$.

Proof. It follows by application of Lemma 4.3 between votes in both bulletin boards BB_0 and BB_1, illustrated in Tables 3 and 4. □

Suppose an adversary colludes with η dishonest voters and attempts to learn the plaintext values of the votes cast by the non-colluding, honest voters. We make use of the indistinguishability relation between bulletin boards to prove in the Lemma 4.5 that the adversary may only learn the partial tally of the $n - \eta$ uncompromised votes within an election using VERICONDOR.

Lemma 4.5. *An adversary who colludes with $\eta < n$ voters learns no information other than the partial tally of the $n - \eta$ uncompromised votes.*

Proof. We refer to the honest voters as $\mathcal{H} = \{0, 1, \ldots, n - 1 - \eta\}$ and the set of dishonest voters as $\mathcal{D} = \{n-\eta, n-\eta+1, \ldots, n-1\}$. The adversary will know the votes of the dishonest voters and hence can determine $g_0^{\mathbf{X}_\kappa(i,j)}$ for all $\kappa \in \mathcal{H}$ and $i, j \in C$. Applying Lemma 4.4 by setting $\eta = n - \eta$ gives the adversary a view of the $n - \eta$ encrypted votes of the honest voters. By Lemma 4.4 this view is indistinguishable to another view of encrypted votes with the same partial tally of honest votes. As the adversary cannot distinguish between votes in the two views they cannot possibly learn the plaintext values of individual votes and hence may only learn the partial tally of the $n - \eta$ honest votes. □

5 PERFORMANCE ANALYSIS

We build a prototype implementation of VERICONDOR in Java.[3] We evaluate the system performance both theoretically and empirically using the Java Microbenchmark Harness (JMH) [41]. The theoretical analysis is found matching the empirical results.

5.1 Runtime Analysis

For our theoretical estimate of the runtime of VERICONDOR, we mainly consider the number of exponentiations that must be performed as exponentiation is the most expensive operation in our system. To analyse the generation of individual ballots B_k, we must determine the number of exponentiations required for generating b_k, Y_k and $P_{WF}\{B_k\}$. For generating b_k and Y_k, each requires n^2 exponentiations. For generating $P_{WF}\{B_k\}$, recall that $P_{WF}\{B_k\}$ consists of five conjunctive statements (C1) through to (C5); we implement the generation $P_{WF}\{B_k\}$ by generating subproofs for each of the five conjunctive statements in our prototype implementation. Hence we analyse the number of exponentiations required by $P_{WF}\{B_k\}$ by summing the estimated cost of generating each of the five subproofs corresponding to each of the five conjunctive statements. Among them, (C1) requires n^2 exponentiations to generate. (C2) only requires n exponentiations to generate as it is defined only over each $i \in C$. (C3) requires generation of a disjunctive NIZKP and requires $3n^2$ exponentiations to generate. (C4) requires the conjunction of two disjunctive NIZKPs over $n^2 - n$ elements in b_k (note that elements in the diagonal of b_k are excluded since they are equal to 1). (C4) requires $6(n^2 - n)$ exponentiations. (C5) requires $4n^2$ exponentiations; it should also be noted that the runtime of computing $\mathbf{X}_k^{\Sigma}$ and b_k^{Π} is $O(n^2)$ and hence these computations provide a negligible overhead when generating $P_{WF}\{B_k\}$. The total number of exponentiations required to generate $P_{WF}\{B_k\}$ is hence $14n^2 - 5n$. The total number of exponentiations required to generate each individual ballot B_k is then $16n^2 - 5n$.

The same reasoning can be applied to verifying individual ballots; each individual ballot B_k requires $19n^2 - 5n$ exponentiations to verify $P_{WF}\{B_k\}$. Our analysis shows that VERICONDOR requires a quadratic number of exponentiations in terms of the number of candidates n for generating and verifying ballots. This $O(n^2)$ complexity is close to optimal as the size of each $\mathbf{V}_k$ is also quadratic in terms of the number of candidates n.

[3]The source code of our implementation can be found at: https://osf.io/g2hfm/?view_only=48fc2c9412e14d1c84b7ae7171b048dd.

$g_0^{X_{\mathbf{0}}(0,0)} g_0^{V_{\mathbf{0}}(0,0)}$	$g_0^{X_{\mathbf{0}}(0,1)} g_0^{V_{\mathbf{0}}(0,1)}$	$\cdots$	$g_0^{X_{\mathbf{0}}(0,n-1)} g_0^{V_{\mathbf{0}}(0,n-1)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$g_0^{X_{\mathbf{0}}(n-1,0)} g_0^{V_{\mathbf{0}}(n-1,0)}$	$g_0^{X_{\mathbf{0}}(n-1,1)} g_0^{V_{\mathbf{0}}(n-1,1)}$	$\cdots$	$g_0^{X_{\mathbf{0}}(n-1,n-1)} g_0^{V_{\mathbf{0}}(n-1,n-1)}$
	$\vdots$		
$g_0^{X_{\boldsymbol{\eta-1}}(0,0)} g_0^{V_{\boldsymbol{\eta-1}}(0,0)}$	$g_0^{X_{\boldsymbol{\eta-1}}(0,1)} g_0^{V_{\boldsymbol{\eta-1}}(0,1)}$	$\cdots$	$g_0^{X_{\boldsymbol{\eta-1}}(0,n-1)} g_0^{V_{\boldsymbol{\eta-1}}(0,n-1)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$g_0^{X_{\boldsymbol{\eta-1}}(n-1,0)} g_0^{V_{\boldsymbol{\eta-1}}(n-1,0)}$	$g_0^{X_{\boldsymbol{\eta-1}}(n-1,1)} g_0^{V_{\boldsymbol{\eta-1}}(n-1,1)}$	$\cdots$	$g_0^{X_{\boldsymbol{\eta-1}}(n-1,n-1)} g_0^{V_{\boldsymbol{\eta-1}}(n-1,n-1)}$

Table 3: Bulletin Board BB_0.

$g_0^{X_{\mathbf{0}}(0,0)} g_0^{V'_{\mathbf{0}}(0,0)}$	$g_0^{X_{\mathbf{0}}(0,1)} g_0^{V'_{\mathbf{0}}(0,1)}$	$\cdots$	$g_0^{X_{\mathbf{0}}(0,n-1)} g_0^{V'_{\mathbf{0}}(0,n-1)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$g_0^{X_{\mathbf{0}}(n-1,0)} g_0^{V'_{\mathbf{0}}(n-1,0)}$	$g_0^{X_{\mathbf{0}}(n-1,1)} g_0^{V'_{\mathbf{0}}(n-1,1)}$	$\cdots$	$g_0^{X_{\mathbf{0}}(n-1,n-1)} g_0^{V'_{\mathbf{0}}(n-1,n-1)}$
	$\vdots$		
$g_0^{X_{\boldsymbol{\eta-1}}(0,0)} g_0^{V'_{\boldsymbol{\eta-1}}(0,0)}$	$g_0^{X_{\boldsymbol{\eta-1}}(0,1)} g_0^{V'_{\boldsymbol{\eta-1}}(0,1)}$	$\cdots$	$g_0^{X_{\boldsymbol{\eta-1}}(0,n-1)} g_0^{V'_{\boldsymbol{\eta-1}}(0,n-1)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$g_0^{X_{\boldsymbol{\eta-1}}(n-1,0)} g_0^{V'_{\boldsymbol{\eta-1}}(n-1,0)}$	$g_0^{X_{\boldsymbol{\eta-1}}(n-1,1)} g_0^{V'_{\boldsymbol{\eta-1}}(n-1,1)}$	$\cdots$	$g_0^{X_{\boldsymbol{\eta-1}}(n-1,n-1)} g_0^{V'_{\boldsymbol{\eta-1}}(n-1,n-1)}$

Table 4: Bulletin Board BB_1.

We now consider the performance of computing the tally verification equations (TV1) and (TV2). The total number of exponentiations required to verify (TV1) and (TV2) is $3n^2$. These two equations are however defined over the set of confirmed ballots $\mathbb{C}$, hence it is also necessary to consider the number of multiplications required. Computing and $\prod_{k \in \mathbb{C}} b_k(i,j)$ and $\prod_{k \in \mathbb{C}} Y_k(i,j)$ both require $|\mathbb{C}|n^2$ multiplications each. A final n^2 multiplications is required to compute the LHS of (TV1). Hence the total number of multiplications required to verify both (TV1) and (TV2) is $2|\mathbb{C}|n^2 + n^2 = n^2(2|\mathbb{C}| + 1)$. This result shows that the number of multiplications required is a product of n^2 and $|\mathbb{C}|$. This is insignificant as most of the runtime of VERICONDORwill be dominated by the more expensive exponentiation operations.

5.2 Microbenchmarks

We recorded the average time to run the ballot creation, verification of **T** and **S** and verification of $P_{WF}\{B_k\}$ within our proof-of-concept implementation using JMH [41]. Figure 3 shows the results when using a 2048-bit p, g_0 and g_1 and a 224-bit q for 112-bit security. Figure 4 shows the results when using a 3072-bit p, g_0 and g_1 and a 256-bit q for 128-bit security.

Our microbenchmark results are consistent with our theoretical runtime analysis. The most theoretically demanding computation in terms of the number of exponentiations required is the verification of $P_{WF}\{B_k\}$; this computation is the one which takes the longest average time across all candidates in Figures 3 and 4. Generation of individual ballots is the next most demanding computation, however not by a large margin. The difference between ballot verification and ballot generation in terms of the number of exponentiations required is $3n^2$. This is reflected in Figures 3 and 4 by the curves for ballot creation appearing slightly below those for the verification of $P_{WF}\{B_k\}$. Our microbenchmark results for tally verification are performed using a fixed number of votes greater than the number of candidates; in each case the number of votes used is 50. Although a large number of multiplications is required for tally verification, it is clear from Figures 3 and 4 that exponentiation is much more demanding than multiplication and hence exponentiation has the biggest impact on the average time of the computations performed by VERICONDOR.

Comparing Figures 3 and 4 directly, it is clear that each of the three benchmarked operations run almost twice as fast when using parameters for 112-bit security compared to using parameters for 128-bit security. The average times of generating individual ballots for 128-bit security, as shown in Figure 4, can be improved by making use of Elliptic Curve Cryptography (ECC) instead. The specification of our protocol remains unchanged. Additionally, our proof-of-concept implementation can be improved by applying parallelism. Each entry in b_k and Y_k may be generated in parallel. Each of the five conjunctions in $P_{WF}\{B_k\}$ may also be generated and verified in parallel. Applying these improvements to our proof-of-concept implementation would result in a more efficient implementation of VERICONDOR. We plan to address this in future work.

6 RELATED WORK

A number of E2E-verifiable voting schemes have been proposed in the literature. The vast majority of these schemes focus on plurality voting, with few schemes designed for ranked-choice voting. The general mix-net technique [7] may be used to build an E2E-verifiable Condorcet voting system, but this requires a set of TAs to run the mix-net servers. Furthermore, the system can be subject to an Italian attack. Another approach is to make use of DRE-ip [15]

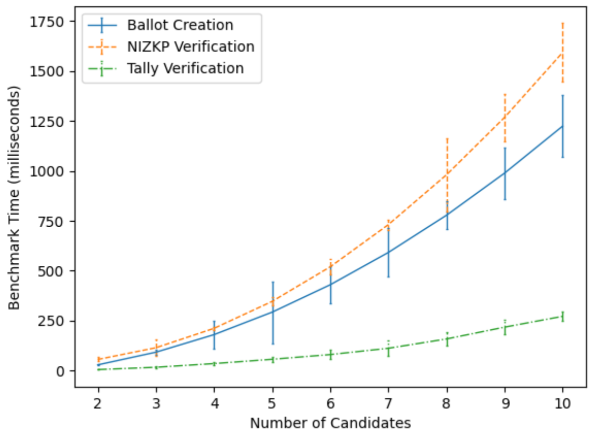

Figure 3: Benchmark results for 112-bit security.

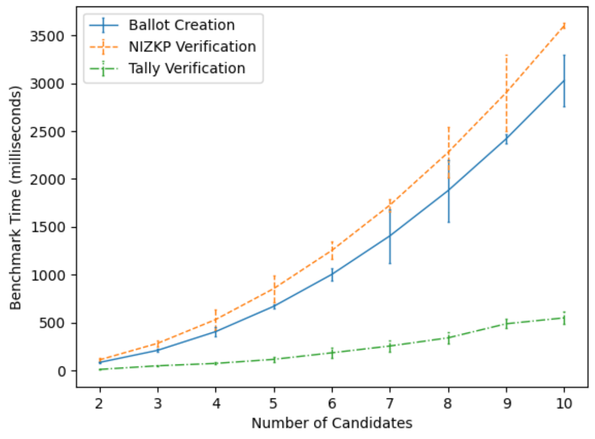

Figure 4: Benchmark results for 128-bit security.

to create a trivial Condorcet voting system whereby each of the n! possible ranked lists will be treated as a candidate in DRE-ip. Whilst straightforward, this alternative approach has exponential complexity and is also potentially subject to an Italian attack.

In the literature, the only prior work that we are aware of on verifiable Condorcet voting is due to Lee and Doi in 2005 [42]. The authors proposed a TA-based *partially* verifiable voting scheme designed specifically to elect a Condorcet winner for an election. Their scheme assumes a trusted voting client, which encrypts the voter's choice honestly. It is possible to use a similar voter-initiated auditing technique [20] to allow the voter to check whether a vote is "cast as intended" without having to trust the voting client, but this depends on how this technique is integrated into the overall voting system, which is not described in their paper. Their scheme involves the encryption and permutation of comparison matrices under an ElGamal cryptosystem, and requires trusted authorities to perform and mixing and decryption operations. Lee and Doi specify the requirement of two different authorities: a tallying authority (TA) for performing the encryption and permutation procedures and a *judging authority* (JA) for performing decryption and determining the Condorcet winner. The computational cost for generating a ballot in their scheme is $O(n^3)$, which is more expensive than the $O(n^2)$ computation cost in our scheme. We note that based on using a comparison matrix to tally votes in a Condorcet voting system, the $O(n^2)$ complexity is probably the best one can hope for.

7 CONCLUSION

In this paper we proposed VERICONDOR: the first E2E-verifiable Condorcet voting system without any tallying authorities. Our system is based on tallying votes in a pairwise comparison matrix and applies a novel method to prove the well-formedness of the matrix with exceptional efficiency. Our system has a computation complexity of $O(n^2)$ per vote where n is the number of candidates. The $O(n^2)$ complexity is close to the best that one may hope for given the use of a $n \times n$ comparison matrix to record the Condorcet vote. The system elects a Condorcet winner when they exist, and has the flexibility to support several Condorcet methods to break a tie and elect an alternative winner in the even that a Condorcet winner does not exist; it is E2E verifiable and guarantees tallying integrity even when the system is completely compromised by a powerful adversary; it protects the secrecy of individual ballots and limits any colluding set of voters to learn nothing more than the total tally and the partial tally of the colluding set. Finally, we presented a prototype implementation and demonstrated the efficiency of the system for practical use. In future work, we plan to extend the system by allowing indifference between candidates in the voting choice (i.e., partial ranking of candidates).

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their very helpful comments. The first author is funded by an EPSRC studentship (No. 2436418). The second and the last authors are funded by Royal Society (ICA\R1\180226) and EPSRC (EP/T014784/1).

REFERENCES

[1] Partha Dasgupta and Eric Maskin. Elections and strategic voting: Condorcet and borda. In *Unpublished slides, UC Irvine Conference on Adaptive Systems and Mechanism Design*, 2010.
[2] H Peyton Young. Condorcet's theory of voting. *The American Political Science Review*, pages 1231–1244, 1988.
[3] Markus Schulze. The schulze method of voting. *arXiv preprint arXiv:1804.02973*, 2018.
[4] Feng Hao and Peter YA Ryan. *Real-World Electronic Voting: Design, Analysis and Deployment.* CRC Press, 2016.
[5] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3):40–46, 2008.
[6] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L Rivest, Peter YA Ryan, Emily Shen, and Alan T Sherman. Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. *EVT*, 8:1–13, 2008.
[7] Peter YA Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *IEEE transactions on information forensics and security*, 4(4):662–673, 2009.
[8] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, et al. Star-vote: A secure, transparent, auditable, and reliable voting system. In

2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13), 2013.

[9] Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. D-demos: A distributed, end-to-end verifiable, internet voting system. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 711–720. IEEE, 2016.

[10] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Demos-2: scalable e2e verifiable elections without random oracles. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 352–363, 2015.

[11] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[12] Ben Adida, Olivier De Marneffe, Olivier Pereira, Jean-Jacques Quisquater, et al. Electing a university president using open-audit voting: Analysis of real-world use of helios. *EVT/WOTE*, 9(10), 2009.

[13] Feng Hao, Matthew N Kreeger, Brian Randell, Dylan Clarke, Siamak F Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. In *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*, 2014.

[14] Feng Hao. Dre-i and self-enforcing e-voting. *Real-World Electronic Voting: Design, Analysis and Deployment*, page 343, 2016.

[15] Siamak F Shahandashti and Feng Hao. Dre-ip: A verifiable e-voting scheme without tallying authorities. In *European Symposium on Research in Computer Security*, pages 223–240. Springer, 2016.

[16] Samiran Bag, Muhammad Ajmal Azad, and Feng Hao. E2e verifiable borda count voting system without tallying authorities. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–9, 2019.

[17] Gerry Mackie. *Democracy defended*. Cambridge University Press, 2003.

[18] Feng Hao, Shen Wang, Samiran Bag, Rob Procter, Siamak F Shahandashti, Maryam Mehrnezhad, Ehsan Toreini, Roberto Metere, and Lana YJ Liu. End-to-end verifiable e-voting trial for polling station voting. *IEEE Security & Privacy*, 18(6):6–13, 2020.

[19] Douglas Robert Stinson and Maura Paterson. *Cryptography: theory and practice*. CRC press, 2018.

[20] Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. *EVT*, 7:14–14, 2007.

[21] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

[22] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[23] Andrew Clausen. Logical composition of zero-knowledge proofs. *Electronic version found in www. cis. upenn. edu/˜ mkearns/teaching/Crypto/zkp-disj. pdf*, 2011.

[24] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[25] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[26] Duncan Black et al. The theory of committees and elections. 1958.

[27] Paul B Simpson. On defining areas of voter choice: Professor tullock on stable voting. *The Quarterly Journal of Economics*, 83(3):478–490, 1969.

[28] Gerald H Kramer. A dynamical model of political equilibrium. *Journal of Economic Theory*, 16(2):310–334, 1977.

[29] Arthur H Copeland. A reasonable social welfare function. Technical report, mimeo, 1951. University of Michigan, 1951.

[30] T Nicolaus Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4(3):185–206, 1987.

[31] Charles Dodgson. A method of taking votes on more than two issues. *The theory of committees and elections*, 1876.

[32] John G Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

[33] H Peyton Young and Arthur Levenglick. A consistent extension of condorcet's election principle. *SIAM Journal on applied Mathematics*, 35(2):285–300, 1978.

[34] John Bartholdi, Craig A Tovey, and Michael A Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and welfare*, 6(2): 157–165, 1989.

[35] Donald G. Saari and Vincent R. Merlin. The copeland method: I.: Relationships and the dictionary. *Economic Theory*, 8(1):51–76, 1996. ISSN 09382259, 14320479. URL http://www.jstor.org/stable/25054952.

[36] Dat-Dao Nguyen. Using social choice function vs. social welfare function to aggregate individual preferences in group decision support systems. *International Journal of Management & Information Systems (IJMIS)*, 18(3):167–172, 2014.

[37] Thomas M Zavist and T Nicolaus Tideman. Complete independence of clones in the ranked pairs rule. *Social Choice and Welfare*, 6(2):167–173, 1989.

[38] Josh Daniel Cohen Benaloh. Verifiable secret-ballot elections. 1989.

[39] Kaoru Kurosawa and Ryo Nojima. Simple adaptive oblivious transfer without random oracle. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 334–346, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10366-7.

[40] Boaz Barak. Lecture 4-computational indistinguishability, pseudorandom generators, 2007.

[41] Java microbenchmark harness (jmh). https://openjdk.java.net/projects/code-tools/jmh/. Accessed: 2021-05-14.

[42] Yoon Cheol Lee and Hiroshi Doi. On the security of condorcet electronic voting scheme. In *International Conference on Computational and Information Science*, pages 33–42. Springer, 2005.

APPENDIX

A NIZKP FOR FINAL CONJUNCTION

From Theorem 2.1, we know that a pairwise comparison matrix $\mathbf{V}$ represents one of the $n!$ votes for an n-candidate election iff $\mathbf{V}^{\Sigma}$, the vector of row sums of $\mathbf{V}$, is a permutation of the set of candidates C. We make use of this to ensure that B_k encodes a valid pairwise comparison matrix. First consider computing the value of $\prod_{j=0}^{n-1} b_k(i,j)$ for a fixed i. We reduce this as follows:

$$\begin{aligned}\prod_{j=0}^{n-1} b_k(i,j) &= g_0^{X_k(i,0)} g_0^{V_k(i,0)} \times \ldots \times g_0^{X_k(i,n-1)} g_0^{V_k(i,n-1)} \\ &= g_0^{X_k(i,0)\,+\,\ldots\,+\,X_k(i,n-1)\,+\,V_k(i,0)\,+\,\ldots\,+\,V_k(i,n-1)} \\ &= g_0^{X_k^{\Sigma}(i)\,+\,V_k^{\Sigma}(i)}\end{aligned}$$

Hence we have $\prod_{j=0}^{n-1} b_k(i,j) = g_0^{X_k^{\Sigma}(i) + V_k^{\Sigma}(i)}$. For brevity we refer to the result of $\prod_{j=0}^{n-1} b_k(i,j)$ or $g_0^{X_k^{\Sigma}(i) + V_k^{\Sigma}(i)}$ as $b_k^{\Pi}(i)$. To prove that $\mathbf{V}_k^{\Sigma}$ is a permutation of C, it is sufficient to prove the following:

$$\bigwedge_{c \in C} c \in \mathbf{V}_k^{\Sigma}$$

These relations prove that each $c \in C$ is an element of the vector $\mathbf{V}_k^{\Sigma}$. Given that C consists of only distinct values, and the cardinality of C is equal to the length of $\mathbf{V}_k^{\Sigma}$, it follows that C must be a permutation of $\mathbf{V}_k^{\Sigma}$ [16] and equivalently that $\mathbf{V}_k^{\Sigma}$ is a permutation of C. These relations are equivalent to the following logical statement adapted from Bag et al. [16]:

$$\bigvee_{i \in C} b_k^{\Pi}(i) = g_0^{X_k^{\Sigma}(i) + j}$$

For all $j \in [0, n-1]$, exactly one of the above disjunctions should hold [16]. Suppose that, for a fixed j, the m^{th} disjunction is true, i.e. we have $b_k^{\Pi}(m) = g_0^{X_k^{\Sigma}(m)} g_0^{j}$. The prover (the DRE-machine) chooses a random $u_m \in_R \mathbb{Z}_q^*$ and computes $t'_m = g_0^{u_m}$. The prover then chooses $r_0, r_1, \ldots, r_{m-1}, r_{m+1}, r_{m+2}, \ldots, r_{n-1} \in_R \mathbb{Z}_q^*$ and also $c_0, c_1, \ldots, c_{m-1}, c_{m+1}, c_{m+2}, \ldots, c_{n-1} \in_R \mathbb{Z}_q^*$ and computes:

$$\forall i \in [0, n-1] - \{m\} : t'_i = g_0^{r_i} \left(\frac{b_k^{\Pi}(i)}{g_0^{j}} \right)^{c_i} \pmod{p}$$

Let the grand challenge be c, where:

$$c = H(k, j, b_k^{\Pi}, t'_0, t'_1, \ldots, t'_{n-1})$$

Here we assume that $H(\cdot)$ is a cryptographically-secure hash function. The prover then computes:

$$c_m = c - \sum_{i \in [0,n-1]-\{m\}} c_i \pmod q$$

$$r_m = u_m - c_m X_k^{\Sigma}(m) \pmod q$$

The subproof is then successful if the following n verification equations are satisfied:

$$\forall i \in [0, n-1] : g_0^{r_i} = \frac{t_i'}{\left(\frac{b_k^{\Pi}(i)}{g_0^j}\right)^{c_i}} \pmod p$$