VERICONDOR: End-to-End Verifiable Condorcet Voting with support for Strict Preference and Indifference

LUKE HARRISON, University of Warwick, United Kingdom
SAMIRAN BAG, University of Warwick, United Kingdom
HANG LUO, Peking University, China
FENG HAO, University of Warwick, United Kingdom

Condorcet voting is widely regarded as one of the most important voting systems in social choice theory. However, it has seen little adoption in practice, due to complex tallying and the need to break ties when there is a Condorcet cycle. Several online Condorcet voting systems have been developed to perform digital tallying and tie-breaking procedures, but they require voters to completely trust the server. Additionally, many end-to-end (E2E) verifiable e-voting systems require trustworthy authorities to perform complex decryption and tallying operations. We propose VERICONDOR, the first E2E verifiable Condorcet e-voting system without tallying authorities. VERICONDOR allows a voter to fully verify the tallying integrity by themselves while providing strong protection of ballot secrecy. We present novel zero-knowledge proof techniques to prove the well-formedness of an encrypted ballot with exceptional efficiency. VERICONDOR supports ranking candidates with strict preference, as well as indifference. The computational cost is exceptionally efficient for strict preferences at $O(n^2)$ per ballot for n candidates, whilst remaining practical for indifferences at $O(n^3)$. In the case of ties, we show how to apply known Condorcet methods to break them in a publicly verifiable manner. Finally, we present a proof of concept implementation and evaluate its performance.

CCS Concepts: • **Security and privacy → Cryptography**.

ACM Reference Format:
Luke Harrison, Samiran Bag, Hang Luo, and Feng Hao. 2024. VERICONDOR: End-to-End Verifiable Condorcet Voting with support for Strict Preference and Indifference. 1, 1 (June 2024), 31 pages.

1 INTRODUCTION

Many different electoral systems have been proposed in the literature for deciding upon a winner for a contest or an election. The simplest, and often most-applied electoral system in practice, is the *plurality method*, whereby the participating voters in an election each cast a vote for one candidate and the candidate who receives the most votes is elected as the winner. Within a 2-candidate election, the basic principle of requiring a winner to receive the most votes is generally accepted. For larger elections however, the plurality method is known to be susceptible to *vote splitting*: the presence of multiple similar candidates may increase the probability that a dissimilar candidate wins the election. In the setting of a US presidential election, consider a comparison between Trump, Clinton and Bloomberg. Suppose a voter prefers Bloomberg to Clinton and Clinton to Trump. If Bloomberg has no chance of winning the election, then the voter who picks Bloomberg as their most-preferred choice would essentially abdicate their influence on the choice between Clinton and Trump [1]. This may result in the least desirable candidate, i.e. Trump, winning the election.

Improvements to the plurality method can be made by allowing voters to *rank* each potential candidate and by using information from each ranking when determining the contest or election winner. One of the most important ranking-based electoral systems in social choice theory is *Condorcet voting*, which was first proposed by Marquis de Condorcet (1743 – 1794). According to Condorcet's criterion, the winner of an election is the candidate who defeats every other candidate by a pairwise majority [2]. This candidate is considered the most socially-optimal choice and is

Authors' addresses: 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.

referred to as the *Condorcet winner*. Conversely, a candidate who loses pairwise to every other candidate is the *Condorcet loser*.

We consider an example set of rankings in Table 1, building upon the considered election between Trump, Clinton and Bloomberg [1]. In this example, 40% of voters prefer Trump to Bloomberg, who in turn is preferred to Clinton. However, both Clinton and Bloomberg would defeat Trump in a head-to-head contest, as 60% of voters prefer either Clinton or Bloomberg 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.

Vote Share	$\mathbf{1}^{st}$	$\mathbf{2}^{nd}$	$\mathbf{3}^{rd}$
40%	Trump	Bloomberg	Clinton
35%	Clinton	Bloomberg	Trump
25%	Bloomberg	Clinton	Trump

Table 1. An example to illustrate vote splitting.

Condorcet voting is preferred when electing a socially optimal choice is considered desirable. This voting method has been utilised in practice by multiple groups and organisations including the Wikimedia Foundation, Debiant, Gentoo, Ubuntu, K Desktop Environment, the pirate party of Sweden, Open-Stack, ICANN, ACM, IEEE, USENIX and Google for internal decisions and polling [3]. Additionally, the third-largest party in the United States, the Libertarian Party of Washington (LPWA) (as of March 2021 [4]), utilises Condorcet voting as part of their internal voting rules [5]. Online elections using Condorcet methods are also available. The Condorcet Internet Voting Service (CIVS)[1] is free and maintained by Andrew Myers of Cornell University. OpaVote[2] is another online service that supports a variety of Condorcet methods, albeit as a paid commercial service.

Despite its importance to social choice theory, Condorcet voting has seen little adoption in wider applications due to several difficulties. Firstly, the voting and counting procedures are more complex compared to those for the plurality method. Voters are required to construct their ballots by ranking candidates in an election. For large elections, construction and counting of these ballots when cast on paper can be tedious and prone to human error. Additionally, it can be difficult for a voter to decide upon a suitable position for every candidate in their ranking; a voter's opinion may be *indifferent* between two candidates or a voter may have strong support for a few candidates and little or no support for the remaining candidates. Riker [6] highlights that economists who rank continuous phenomena like money may find allowing indifference to be useful in their rankings, while political scientists who rank discrete phenomena such as political alternatives may prefer to only use strict preference in their rankings. The use of computerised input can help voters in constructing their ballot and the use of digital technologies can help to speed up the counting of ballots, however the usage of these systems also introduces the threat that the digital data can be modified to alter the election result. Secondly, in some elections, a Condorcet winner may not exist. Consider an election between three candidates A, B and C. If A is preferred to B, B is preferred to C and C is preferred to A after counting, then there is a circular relation between all three candidates and there is no Condorcet winner. A few solutions have been proposed to break the tie in the event that a Condorcet winner does not exist [3]. However, there has been limited research on how a tie can be broken in a public, yet privacy-preserving manner. Previous solutions typically focus on specific tie-breaking Condorcet methods facilitated by tallying authorities, such as the papers

[1]https://civs1.civs.us/

[2]https://www.opavote.com/

authored by Hertel et al. [7] and Cortier et al. [8]. In our work, we will address the problem of breaking ties in a public, yet privacy-preserving manner without the need for any tallying authority.

In this paper, we design a fully verifiable Condorcet e-voting system, addressing both problems above. Our design builds on the well-established notion of *end-to-end (E2E) verifiability* [9], which encompasses the following properties.

(1) *Cast as intended* – any voter may verify that their cast vote truly represents their chosen candidate and not another;
(2) *Recorded as cast* – any voter may verify that their vote is recorded and included as part of the tallying process;
(3) *Tallied as recorded* – anyone (including a third-party observer) may verify that the final tally is truly the result of aggregating each recorded vote.

E2E verifiable e-voting systems often require a selected group of authorities who are supposedly trustworthy individuals tasked to perform decryption and tallying operations, as well as sometimes mixing votes using a chain of mix-net servers. We refer to these authorities as *tallying authorities* or TAs. Examples of these solutions include Scantegrity II [10], Prêt à voter [11], DEMOS-2 [12] and Helios [13]. However, in practice, choosing and managing the TAs has proved to be particularly difficult [14]. These concerns led to the advancement of Self-Enforcing E-Voting systems, often abbreviated to SEEV systems [15]. These systems realize E2E verifiability without involving any TAs. Examples include DRE-i [16], DRE-ip [17] and DRE-Borda [18]. 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. Existing online Condorcet voting systems (e.g., CIVS and OpaVote) that provide voting services to real-world voters are not E2E verifiable; if the voting server is compromised, the tallying integrity of an election will be completely lost. A major challenge in building an E2E verifiable Condorcet voting system, as we will explain below, is to represent and subsequently prove that an encrypted ballot is in a format permissible for counting without involving any tallying authorities.

In this paper we propose VERICONDOR, the first E2E-verifiable Condorcet voting system, which supports both strict preference and indifference in the ranking of candidates. Our system does not require any tallying authority. This is inspired by DRE-ip [17], 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. We introduce novel zero-knowledge proof techniques to prove the well-formedness of a Condorcet ballot. 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 TAs, supporting both strict preference and indifference. Our system is exceptionally efficient for strict preferences, incurring only $O(n^2)$ computation for each ballot where n is the number of candidates. For indifferences, the cost per ballot remains practical at $O(n^3)$.
(2) We discuss how to elect an alternative winner in a publicly verifiable manner, in the event that the Condorcet winner does not exist, based on a selection of Condorcet methods.
(3) We present rigorous proofs to show that the VERICONDOR protocol is secure based on the Decision Diffie-Hellman (DDH) assumption in a random oracle model.
(4) We build an open-source proof of concept implementation for the VERICONDOR system and present detailed performance measurements to demonstrate the feasibility of our system.

A preliminary version of this paper was presented at a conference [19]. This journal paper extends the conference version as follows. First, we add support for ranking candidates with *indifference*, while the previous work only allows *strict preference*. Permitting indifference poses a significantly harder challenge than only allowing strict preference in terms of proving the well-formedness of an encrypted ballot, but at the same time, it also makes our system more versatile to support applications with different ranking requirements. Second, for the choice of ranking with strict preference, we improve the earlier work by introducing a new technique that allows us to securely use only half of the matrix instead of a full matrix to record votes. This reduces the computation by half, although the system complexity remains the same. Third, we provide an updated open-source implementation that supports not only strict preference but also indifference. The performance evaluation and security proofs have also been updated accordingly.

2 PRELIMINARIES

We begin with an introduction of Condorcet methods and how these methods represent and tally votes containing strict preferences.

2.1 Condorcet Voting with Strict Preference

Elections using Condorcet methods typically require voters to rank the candidates in order of preference. For simplicity, we discuss ranking with *strict preference* first, and will cover the case of ranking with *indifference* in Section 3.4. For an n-candidate election with candidates belonging to the set $C = \{0, 1, ..., n-1\}$, each vote may be represented as a permutation $\rho = (c_0, c_1, ..., c_{n-1})$ 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 ρ when ρ is read from left-to-right in descending order. We may write this preference as $c_a \succ c_b$. Consider a 3-candidate election with $C = \{A, B, C\}$ (we may assume $A = 0$, $B = 1$ and $C = 2$). A voter may choose the permutation (B, C, A) as their vote, meaning that they prefer candidate B to all other candidates, candidate C to candidate A and candidate A to no other candidates.

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) [9]. Here TAs are required to run the mix-net servers, which is a complex task. Furthermore, the decryption of all votes renders the system vulnerable to an Italian attack [9], in which a voter is coerced to choose an uncommon permutation of candidates so the coercer can verify it in the decrypted votes. The second method is to encode all of the $n!$ possible ranked lists as voter choices in the DRE-ip protocol [17]. 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. 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 is crucial for maintaining E2E verifiability. We refer to a matrix $\mathbf{V} = (v_{ij})$ as a pairwise comparison matrix, if it satisfies the following basic properties:

$$\forall i \in C : v_{ii} = 0 \tag{P$_1$}$$

$$\forall i, j \in C : v_{ij} = 0 \vee v_{ij} = 1 \tag{P$_2$}$$

$$\forall i, j \in C,\ i \neq j : v_{ij} = 0 \Leftrightarrow v_{ji} = 1 \tag{P$_3$}$$

(P_1) simply states that no candidate can be preferred to themselves. (P_2) states that each entry in the matrix $\mathbf{V}$ can only be 0 or 1. (P_3) 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. We emphasize that these properties are *necessary*, but not *sufficient* to ensure that the matrix must encode a valid ballot, because of additional transitivity constraints (e.g., given $i \succ j$ and $j \succ k$, this imply that we must have $i \succ k$). It is possible to exhaustively list out these constraints and apply corresponding zero-knowledge proof techniques to prove that an encrypted matrix contains a valid ballot, but this will be extremely inefficient. We will present an efficient solution to prove the well-formedness of an encrypted ballot in Section 3.3.

Given a permutation ρ 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 ρ. 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 (B, C, A).

$$(B,C,A) \rightarrow \begin{array}{c|ccc} & \text{A} & \text{B} & \text{C} \\ \hline \text{A} & 0 & 0 & 0 \\ \text{B} & 1 & 0 & 1 \\ \text{C} & 1 & 0 & 0 \end{array} \rightarrow \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Fig. 1. Obtaining a comparison matrix from (B, C, A).

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 votes (B, C, A), (B, A, C) and (A, B, C) is given in Figure 2. In this example, candidate B defeats candidate A by two votes to one vote, and also defeats candidate C by three votes to zero votes. Therefore, candidate B is the Condorcet winner.

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

Fig. 2. An illustration of a sum matrix encompassing three votes: (B, C, A), (B, A, C), and (A, B, C).

2.2 Condorcet Cycle

It may be the case that a Condorcet winner does not exist for an election at all. Consider the sum matrix illustrated within Figure 3. In this example, candidate A defeats candidate B, candidate B defeats candidate C, and candidate C defeats candidate A, each by two votes to one vote. There is a tie between each candidate. Such an outcome is often referred to as a *Condorcet cycle* [2].

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

Fig. 3. An illustration of a sum matrix encompassing three votes: (A, B, C), (C, A, B), and (B, C, A).

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 3.6.

3 THE VERICONDOR SYSTEM

We now describe the operation of VERICONDOR. Our system supports two different ranking methods: one that permits strict preference only, and another that also permits indifference. The main difference between these two ranking methods lies in how to prove the well-formedness of an encrypted ballot. For concreteness, we will describe the system in the context of polling station voting. This follows the same setting as in the DRE-ip paper [17]. (However, the underlying protocol can also be implemented for online voting.)

3.1 Requirements and Assumptions

In a supervised polling station environment, we require a mechanism for voters to rank candidates in order of preference or to indicate indifference between candidates. A touch-screen direct recording electronic (DRE) could be used in this case, allowing voters to reorganise candidates to their liking. Although, the method of interaction is largely irrelevant in the context of the wider VERICONDOR system, and accommodations are possible for voters with particular needs, e.g., the use of assistive technologies for voters with disabilities. We leave a complete exploration on the usability and accessibility of a real-world implementation of VERICONDOR for future work.

We assume there is a secure voter registration process before the election and that only eligible voters are permitted to vote. On the election day, we require voters to be properly authenticated at the polling station. After being authenticated, each voter obtains an anonymous voting credential, which can be a one-time passcode or a randomly assigned smart card. With this credential, the voter enters a private voting booth and logs on to a DRE machine to vote. Since the voter ranks candidates through a touch screen, the DRE machine learns the voter's choice by definition. However, the machine does not know the voter's real identity, therefore the voter's privacy is preserved through anonymity. When a voter confirms casting their ballot, the DRE machine aggregates the ballot to a running tally in memory, transforms it into an encrypted version (for storage and voter verification), and finally securely deletes the plain ballot from the memory. (Here, for simplicity, we assume the protocol is implemented on a DRE machine; in practice it can be easily implemented on a server which is linked up with multiple DRE machines as done in the 2019 DRE-ip e-voting trial [20], so that the tallying can be performed at a polling station level or a precinct level.) Each DRE machine is connected to a local printer for printing paper-based receipts for each voter. Like any other E2E voting system [9], VERICONDOR requires a secure, publicly-accessible, and append-only[3] bulletin board (BB) to facilitate both vote auditing and E2E verifiability. All the receipts that are printed on paper are also published on the BB. In the event that a DRE machine is completely compromised by an adversary, we limit the information leakage to the adversary as just the partial tally at the time of compromise. In this scenario, we will show that the attacker is unable to modify the tally without being detected by the public (due to the E2E property of the system).

3.2 A Basic Scheme

We first present a basic scheme that works generally with different ways of ranking candidates, i.e., whether ranking requires *strict preference* or permits *indifference*. Later, we will provide two

[3]In practice, such assumptions can be difficult to realise. Relaxing these is a problem in and of itself, and multiple works (e.g., [21–23]) provide attempts for this.

detailed constructions for the two ranking methods, based on *strict preference* and *indifference*, in Sections 3.3 and 3.5 respectively. Our protocol comprises three phases: setup, voting and tallying.

3.2.1 Setup. 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 $\mathbb{Z}_p^*$. All modular operations are performed with reference to the modulus p unless stated otherwise.

The setup phase involves generating two random generators of $\mathbb{G}_q$, denoted by g_0 and g_1 whose discrete logarithm is unknown. To construct a pair of generators g_0 and g_1, we may first fix g_0 to be any non-identity element in $\mathbb{G}_q$ and compute g_1 based on a one-way hash function with g_0 and public contextual information (e.g. election title, date and candidates) as the input [20]. As part of the setup phase, the election organisers need to decide whether to require ranking under strict preference, or to permit indifference within votes for the chosen n candidates.

3.2.2 Voting. On the election day, a voter is first authenticated at the polling station and obtains an anonymous credential, e.g., a one-time passcode or a random smart card. The voter enters a private voting booth, logs onto the DRE machine with the obtained credential and starts voting.

In VERICONDOR, casting a vote is done in two steps. In the first step, a voter ranks n candidates in their order of preference 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 [24]. 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. (For the simplicity of illustration, here we use a full matrix to represent a vote; in Section 3.3, we will show that if only *strict preference* is allowed, we can optimize the representation by using only half of a matrix). We denote the plaintext pairwise comparison matrix as $\mathbf{V}_k = (v_{ij})_k$ where k is a unique ballot index ($k = 0, 1, 2, \ldots$). The DRE-machine holds two $n \times n$ matrices $\mathbf{S} = (s_{ij})$ and $\mathbf{T} = (t_{ij})$ which are both initialised to $\mathbf{0}_{n,n}$. The matrix $\mathbf{T}$ is used to store the running tally of the votes received so far while the matrix $\mathbf{S}$ is used to store the aggregated random numbers introduced in the encryption process. The system keeps both $\mathbf{S}$ and $\mathbf{T}$ secret until the end of the election. 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:

$$\forall i \in C : (x_{ii})_k = 0 \tag{X_1}$$

$$\forall i, j \in C, i \neq j : (x_{ij})_k \in_R \mathbb{Z}_q^* \tag{X_2}$$

We use $(\in_R)$ to denote uniformly random selection from a set. (X_1) states that the main diagonal of $\mathbf{V}_k$ always consists of 0s. (X_2) 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 $B_k = \langle b_k, Y_k \rangle$ where:

$$\forall i \in C : (b_{ii})_k = 1 \tag{BC_1}$$

$$\forall i, j \in C, i \neq j : (b_{ij})_k = g_0^{(x_{ij})_k} g_0^{(v_{ij})_k} \tag{BC_2}$$

$$\forall i, j \in C : (Y_{ij})_k = g_1^{(x_{ij})_k} \tag{BC_3}$$

In these equations, "BC" refers to *ballot construction*. A ballot is encrypted in a way similar to ElGamal encryption, but no private key (i.e., the discrete logarithm relation between g_0 and g_1)

is available to perform decryption. Instead, the decryption is performed by cancelling out the random factors (x_{ij}) for all ballots in the tallying process, similar to DRE-ip [17]. (BC_1) states that the encryption of a ranking order between the same candidate is just 1. This is because for any candidate i, both $(x_{ii})_k$ and $(v_{ii})_k$ are 0 and hence $(b_{ii})_k$ must equal 1. (BC_2) and (BC_3) 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 [25]):

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

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 of NIZKPs in Section 3.3 for ranking candidates based on strict preference and Section 3.5 for permitting indifference. 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 the BB for public verification [20].)

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 $\mathbf{S} = (s_{ij})$ and $\mathbf{T} = (t_{ij})$ as follows:

$$\forall i, j \in C, k \in \mathbb{C}\colon s_{ij} = s_{ij} + (x_{ij})_k \tag{T_1}$$

$$\forall i, j \in C, k \in \mathbb{C}\colon t_{ij} = t_{ij} + (v_{ij})_k \tag{T_2}$$

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.

3.2.3 Tallying. When the election day finishes, the DRE-machine publishes $\mathbf{S}$ and $\mathbf{T}$ on the BB. The matrix $\mathbf{S}$ holds the sums of all randomness generated per each entry in a pairwise comparison matrix, which is necessary for verification of the final tally. The matrix $\mathbf{T}$ holds the aggregated preferences after voters have cast their ballots and hence is used to determine the final winner of an election. 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:

$$\forall i, j \in C\colon g_0^{s_{ij}} g_0^{t_{ij}} = \prod_{k \in \mathbb{C}} (b_{ij})_k \tag{TV_1}$$

$$\forall i, j \in C\colon g_1^{s_{ij}} = \prod_{k \in \mathbb{C}} (Y_{ij})_k \tag{TV_2}$$

We use "TV" to refer to *tally verification*. (TV_1) and (TV_2) can be performed by anyone with read access to the public BB. One must simply use the published values of $\mathbf{S}$, $\mathbf{T}$, and all confirmed ballots

to compute the left-hand side and right-hand side of each equation, and then compare the results for equality. If the comparison fails, then the election organisers must be notified immediately.

3.3 ZKPs of Well-formedness for Ranking with Strict Preference

When we represent a Condorcet vote with a list of ranked candidates in strict preference 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.

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \end{array} \tag{1}$$

The above matrix satisfies (P_1), (P_2) and (P_3) listed under Section 2.1, but does not represent a valid ranked list. Here, candidate A is preferred to candidate C, candidate C is preferred to candidate B and candidate B is preferred to candidate A. 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 A is preferred to B and B is preferred to C in $\mathbf{V}$, then we expect A to be preferred to C in $\mathbf{V}$. 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:

$$\begin{aligned} i \succ j \wedge j \succ k &\Rightarrow i \succ k \\ i \not\succ j \wedge j \not\succ k &\Rightarrow i \not\succ k \end{aligned} \tag{2}$$

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, let us first consider the transitive relations among three candidates. It is not difficult to count that the number of constraints required is $\binom{n}{3} \times 3! \times 2 = 2n(n-1)(n-2)$. Hertel et al. [7] show that enforcing the transitive relations for any three candidates is sufficient to ensure the transitive relations for $n \geq 3$ candidates. This results in $O(n^3)$ constraints, which is not ideal for elections with multiple candidates.

We propose a 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, as shown in Theorem 3.1.

Theorem 3.1. An $n \times n$ binary matrix $\mathbf{V} = (v_{ij})$ satisfying $v_{ii} = 0$, $v_{ij} = 0 \vee v_{ij} = 1$ and $v_{ij} = 0 \Leftrightarrow v_{ji} = 1$ for all $i, j \in [0, n-1], i \neq j$ 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.

Theorem 3.1 has been proved in our earlier conference version of the paper [19] (see Appendix A). This theorem provides an efficient way to verify the validity of pairwise comparison matrices compared to the approaches that quantify transitivity constraints. Essentially, it allows us to change a complex problem of proving the well-formedness of the comparison matrix to a simpler but equivalent problem of proving that the vector sum is a permutation of C. As an example, consider

the pairwise comparison matrix for the vote $D \succ B \succ A \succ C$ below.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{3}$$

Based on Theorem 3.1, proving the validity of the matrix is equivalent to proving that the vector sum is a permutation of $\{0, 1, 2, 3\}$, for which there are efficient zero-knowledge proof techniques available. Based on this result, we can construct an efficient system to support verifiable Condorcet voting with strict preference, as shown in our earlier paper [19]. The resultant system complexity is $O(n^2)$, which is the best one may hope for given the use of a $n \times n$ matrix to record a vote.

In this journal paper, we present a new theorem (following Theorem 3.1), which leads to a more compact representation of a ballot. The $O(n^2)$ system complexity remains the same, but the computational cost per ballot is reduced by half. This improvement is based on the observation that one does not need to use the full pairwise comparison matrix to determine if a vote is well-formed. In fact, if we examine the upper triangular matrix above the diagonal and the lower triangular matrix below the diagonal, we find that they are mirrors of each other, with any strict preferences swapped. More specifically, given any pairwise comparison matrix $\mathbf{V} = (v_{ij})$ under the assumption that only strict preference is permitted, we have $v_{ij} = 1 - v_{ji}$ for all $i, j \in [0, n-1]$, $i \neq j$. More than half of a ballot is redundant if the ballot is constructed via the encryption of a full $n \times n$ pairwise comparison matrix.

Given a pairwise comparison matrix $\mathbf{V}$, we refer to the upper triangular of this matrix as $\mathbf{U} = (u_{ij})$. We have $u_{ij} = 1$ if $i \succ j$ in ρ and $u_{ij} = 0$ if $j \succ i$ in ρ. Note that $\mathbf{U}$ consists of $\frac{1}{2}n\,(n-1)$ entries. We may then rewrite $\left(\sum_{j=0}^{n-1} v_{ij}\right)_{i=0}^{n-1}$ as the following.

$$\left(\sum_{j=0}^{n-1} v_{ij}\right)_{i=0}^{n-1} = \left(\sum_{i=c+1}^{n-1} u_{ci} + \sum_{i=0}^{c-1} 1 - u_{ic}\right)_{c=0}^{n-1} = \left(\left[\sum_{i=c+1}^{n-1} u_{ci}\right] + c - \left[\sum_{i=0}^{c-1} u_{ic}\right]\right)_{c=0}^{n-1}$$

From this result, we form a corollary from Theorem 3.1 below.

Corollary 3.1. A triangular matrix $\mathbf{U} = (u_{ij})$ encodes one of the $n!$ possible votes for an n-candidate election, under the assumption that only strict preferences are permitted in the election, iff $\left(\left[\sum_{i=c+1}^{n-1} u_{ci}\right] + c - \left[\sum_{i=0}^{c-1} u_{ic}\right]\right)_{c=0}^{n-1}$ is a permutation of the set of candidates C.

Based on Corollary 3.1, we can compute the vector sum of the matrix in Eq. (3) to be $(1, 2, 0, 3)$, by using only a triangular matrix instead of a full one. Without loss of generality, we use the upper triangular matrix. Performing verification for pairwise comparison matrices runs in time $O(n^2)$, regardless of whether Theorem 3.1 or Corollary 3.1 is used. Both greatly simplify the ZKPs in VERICONDOR as we will explain. The benefit of using the latter however is that it results in the construction of smaller ballots with less complex ZKPs. These smaller ballots and ZKPs are more efficient to construct and verify, and less time is needed for a voter or a third party to verify that a ballot has been recorded and included in the final tally.

It is largely the same to generate individual ballots and construct and verify the tallies when Corollary 3.1 is used in place of Theorem 3.1. Let $\mathbf{U}_k = (u_{ij})_k$ denote the triangular matrix for a voter $k \in K$. The major difference is that one must substitute $(u_{ij})_k$ for $(v_{ij})_k$ in (BC_2), (BC_3), (T_1) and (T_2). It is also possible to make use of the upper triangular matrices for $\mathbf{X}_k$, $\mathbf{T}$ and $\mathbf{S}$ within (X_2),

(BC_2), (BC_3), (T_1), (T_2), (TV_1) and (TV_2). Equations (X_1) and (BC_1) can be omitted when Corollary 3.1 is used as the diagonal is not included within $\mathbf{U}_k$.

To verify the well-formedness of triangular matrices in encrypted ballots, we must verify that each entry in $\mathbf{U}_k$ is either a 0 or a 1 and that the vector $\left(\left[\sum_{i=c+1}^{n-1}(u_{ci})_k\right] + c - \left[\sum_{i=0}^{c-1}(u_{ic})_k\right]\right)_{c=0}^{n-1}$ is a permutation of C. For the latter, consider the following equivalence.

$$g_0^c \cdot \frac{\prod_{i=c+1}^{n-1}(b_{ci})_k}{\prod_{i=0}^{c-1}(b_{ic})_k} = g_0^{\sum_{i=c+1}^{n-1}(x_{ci})_k - \sum_{i=0}^{c-1}(x_{ic})_k} \cdot g_0^{\left[\sum_{i=c+1}^{n-1}(u_{ci})_k\right] + c - \left[\sum_{i=0}^{c-1}(u_{ic})_k\right]}$$

It is then sufficient to prove the following (based on Bag et al. [18]).

$$\bigwedge_{c' \in C} c' \in \left(\left[\sum_{i=c+1}^{n-1}(u_{ci})_k\right] + c - \left[\sum_{i=0}^{c-1}(u_{ic})_k\right]\right)_{c=0}^{n-1}$$

These relations are equivalent to the following logical statement.

$$\bigvee_{c \in C} g_0^c \cdot \frac{\prod_{i=c+1}^{n-1}(b_{ci})_k}{\prod_{i=0}^{c-1}(b_{ic})_k} = g_0^{\sum_{i=c+1}^{n-1}(x_{ci})_k - \sum_{i=0}^{c-1}(x_{ic})_k} \cdot g_0^{\mathcal{J}}$$

For all $\mathcal{J} \in C$, exactly one of the above disjunctions should hold. We make use of the above logical statement to form a conjunctive ZKP that proves whether a triangular matrix is well-formed when encrypted. We denote this ZKP as $P_{WF}\{B_k = \langle b_k, Y_k\rangle\}$. Its definition is given below.

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

$$P_K\Bigg\{(x_{ij})_k : \left(\left(\log_{g_0}(u_{ij})_k = \log_{g_1}(Y_{ij})_k\right) \vee \left(\log_{g_0}(u_{ij})_k / g_0 = \log_{g_1}(Y_{ij})_k\right)\right) \tag{C_1}$$

$$\wedge \bigvee_{c \in C} \log_{g_0}\left(g_0^c \cdot g_0^{-\mathcal{J}} \cdot \frac{\prod_{i=c+1}^{n-1}(b_{ci})_k}{\prod_{i=0}^{c-1}(b_{ic})_k}\right) = \log_{g_1}\left(\frac{\prod_{i=c+1}^{n-1}(Y_{ci})_k}{\prod_{i=0}^{c-1}(Y_{ic})_k}\right) \Bigg\} \tag{C_2}$$

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 $P_K\{(x_{cc'})_k\}$ consisting of two conjunctive statements labelled (C_1) and (C_2) above. We adopt the ZKP proposed by Bag et al. [18] 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. We note that the complexity of the scheme proposed by Bag et al. may be improved to $O(n \log n)$, e.g., by using BulletProof [26], however this will not change the overall $O(n^2)$ complexity for our protocol, which is determined by the triangular matrix of size $\frac{1}{2}n(n-1)$. These ZKPs are then made non-interactive by applying the Fiat-Shamir heuristic [27]. Same as in DRE-ip [17], we require including the unique ballot index k into the hash function to bind the proof to each ballot. This prevents the "replay attack" as reported on Helios 2.0 [28].

Statement (C_1) is straightforward to implement. In Appendix B, we detail the ZKP necessary for the non-trivial statement (C_2). This statement makes use of Corollary 3.1. If Theorem 3.1 is used instead, one must instead provide a ZKP for the statement $\bigvee_{i \in C} \prod_{j=0}^{n-1}(b_{ij})_k = g_0^{\sum_{j=0}^{n-1}(x_{ij})_k + \mathcal{J}}$, alongside ZKPs that specify $v_{ij} = 1 - v_{ji}$.

3.4 Support for Indifference

So far we have assumed that a voter ranks candidates in *strict preference*, but sometimes a voter may be *indifferent* between two alternatives if they cannot decide whether to rank one alternative above another. Previously, we introduced votes as a single permutation ρ that encodes the strict preference relation ($>$) implicitly. To model indifference, we introduce an additional vector v where

$\forall i \in [0, n-2] : v_i \in \{(\succ), (\sim)\}$. The vector v defines the relation between successive elements in ρ, with $(\succ)$ representing a strict preference between two candidates and $(\sim)$ representing indifference between two candidates.

Let ρ_x denote the candidate at position x in ρ. Suppose we have $\rho_x = i$ and $\rho_z = j$ with $i, j \in C$ and $x < z$. A candidate i is then *preferred* to candidate j in the vote (ρ, v) if $\exists y \in [x, z) : v_y = (\succ)$; again, we may write this preference as $i \succ j$. We say that candidate i is *indifferent* to candidate j if $\forall y \in [x, z) : v_y = (\sim)$; we may write this indifference as $i \sim j$ or $j \sim i$. Consider the set of candidates $C = \{A, B, C, D\}$; again, we assume $A = 0$, $B = 1$, $C = 2$ and $D = 3$. Then one example of a vote is the pair $\langle (B, A, C, D), (\succ, \sim, \succ) \rangle$. In this vote, B is preferred to all other candidates, A and C are indifferent and both A and C are preferred to D. We may also write this more succinctly as $B \succ A \sim C \succ D$. The total number of possible votes in this formalisation is the same as the number of weak orderings of n possible items (also known as the Fubini number or ordered Bell number), expressed as $T(n) = \sum_{m=1}^{n} m! \left\{ {n \atop m} \right\}$ where $\left\{ {n \atop m} \right\} = \frac{1}{m!} \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m - i)^n$ denotes the Stirling numbers of the second kind [29].

We may also make use of the pairwise comparison matrix scheme to tally votes containing indifference. Given a vote (ρ, v), we define the associated pairwise comparison matrix $\mathbf{V} = (v_{ij})$ *with indifferences* as follows.

$$\forall i \in [0, n-1] : v_{ii} = 0 \tag{S_1}$$

$$\forall i, j \in [0, n-1], i \neq j : v_{ij} = 0 \vee v_{ij} = 1 \tag{S_2}$$

$$\forall x, z \in [0, n-1], x < z, \rho_x = i, \rho_z = j : \exists y \in [x, z) : v_y = (\succ) \Rightarrow v_{ij} = 1 \wedge v_{ji} = 0 \tag{S_3}$$

$$\forall x, z \in [0, n-1], x < z, \rho_x = i, \rho_z = j : \forall y \in [x, z) : v_y = (\sim) \Rightarrow v_{ij} = 1 \wedge v_{ji} = 1 \tag{S_4}$$

The idea that no candidate may be preferred or indifferent to themselves is stated by (S_1). The condition that each entry in the matrix can only be a 0 (representing a pairwise defeat) or a 1 (for strict preference or indifference) is expressed by (S_2). Statements (S_3) and (S_4) define the construction of a pairwise comparison matrix from a given vote (ρ, v). (S_3) states that if candidate i is preferred to candidate j in a vote (ρ, v), then this strict preference is reflected by candidate i defeating candidate j and candidate j losing to candidate i in $\mathbf{V}$. (S_4) states that if candidates i and j are indifferent in a vote (ρ, v), then this indifference is reflected by candidates i and j pairwise defeating each other in $\mathbf{V}$. Figure 4 illustrates this construction for the vote $B \succ A \sim C \succ D$. Similar to the earlier example of the matrix with *strict preference*, these four properties do not consider transitivity constraints, and hence are not sufficient to ensure a given matrix must encode a valid ballot; more details on enforcing the well-formedness of a matrix will be presented in Section 3.5.

$B \succ A \sim C \succ D \quad \rightarrow$

	A	B	C	D
A	0	0	1	1
B	1	0	1	1
C	1	0	0	1
D	0	0	0	0

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

Fig. 4. Obtaining a pairwise comparison matrix with indifferences from $B \succ A \sim C \succ D$.

We may again make use of matrix addition to construct the sum matrix *permitting indifference*. Consider the votes $B \succ A \sim C \succ D$, $C \sim B \succ A \succ D$ and $D \succ B \sim C \sim A$. The process of constructing the sum matrix composing these three votes is basically the same as tallying votes of *strict preference* (Fig. 2) and is illustrated by Figure 5.

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

Fig. 5. Construction of the sum matrix with votes $B \succ A \sim C \succ D$, $C \sim B \succ A \succ D$, and $D \succ B \sim C \sim A$.

3.5 ZKPs of Well-formedness for Ranking With Indifference

We now propose a solution to well-formedness verification for pairwise comparison matrices, assuming that indifference is permitted. Unlike the case of *strict preference*, when *indifference* is permitted, there is no longer symmetry in the comparison matrix. As a result, we must use a full matrix to represent the vote. We say that a matrix is *well-formed under indifference* or *valid under indifference* if it represents one of the $T(n)$ possible votes for an n-candidate election that permits indifference (be reminded that $T(n) = \sum_{m=1}^{n} m! \left\{ {n \atop m} \right\}$ from Section 3.4). A matrix is said to be *malformed under indifference* or *invalid under indifference* if it does not satisfy each of (S_1), (S_2), (S_3) and (S_4), or the matrix breaks a constraint. In addition to the two constraints from Eq. (2), we require the following five constraints between i, j, k satisfying $i, j, k \in C \wedge i \neq j \neq k$.

$$\begin{aligned} i \sim j \wedge j \succ k &\Rightarrow i \succ k \\ i \sim j \wedge j \not\succ k &\Rightarrow i \not\succ k \\ i \succ j \wedge j \sim k &\Rightarrow i \succ k \\ i \not\succ j \wedge j \sim k &\Rightarrow i \not\succ k \\ i \sim j \wedge j \sim k &\Rightarrow i \sim k \end{aligned} \tag{4}$$

We require more constraints to satisfy the case with indifference, although the scaling of these constraints remains $O(n^3)$. More importantly however, Theorem 3.1 does not hold for matrices containing indifferences. Theorem 3.1 provides an efficient method for verifying well-formedness of strict preferences, which we desire to use within VERICONDOR. In the following, we show how to reduce the computational burden of permitting indifferences.

It is straightforward to verify that a matrix satisfies both (S_1) and (S_2) once encrypted. It is less straightforward to verify (S_3) and (S_4), or equivalently, that all transitivity constraints hold for a matrix once encrypted. We propose an augmentation to VERICONDOR which adds an additional matrix to a vote. Now, a vote consists of a pairwise comparison matrix $\mathbf{V}_k = (v_{ij})_k$ as well as a new *indifference matrix* $\mathbf{V}_k^{\mathcal{I}} = \left(v_{ij}^{\mathcal{I}}\right)_k$. The purpose of $\mathbf{V}_k^{\mathcal{I}}$ is to state all indifferences for a particular voter. We define $\mathbf{V}_k^{\mathcal{I}}$ through the following properties:

$$\forall i \in C \colon v_{ii}^{\mathcal{I}} = 0 \tag{$P_1^{\mathcal{I}}$}$$

$$\forall i, j \in C,\ i \neq j \colon v_{ij}^{\mathcal{I}} = 0 \vee v_{ij}^{\mathcal{I}} = 1 \tag{$P_2^{\mathcal{I}}$}$$

$$\forall i, j \in C,\ i \neq j \colon v_{ij} = 0 \vee v_{ij}^{\mathcal{I}} = 0 \tag{$P_3^{\mathcal{I}}$}$$

($P_1^{\mathcal{I}}$) and ($P_2^{\mathcal{I}}$) mirror the properties (P_1) and (P_2). The most important property here is ($P_3^{\mathcal{I}}$), which states that $v_{ij}^{\mathcal{I}} = 0$ holds when $v_{ij} = 1$. This allows specification of an indifference between i and j by setting $v_{ji}^{\mathcal{I}} = 1$; if $v_{ij} = 1$, then $v_{ji} = 0$ and $v_{ij} + v_{ij}^{\mathcal{I}} = v_{ji} + v_{ji}^{\mathcal{I}} = 1$, or $i \sim j$. We assume each vote for a voter k is transformed into the two matrices $\mathbf{V}_k$ and $\mathbf{V}_k^{\mathcal{I}}$ when it is cast. Then, the combined pairwise comparison matrix with indifferences is computed as $\mathbf{V}_k + \mathbf{V}_k^{\mathcal{I}}$.

Whilst the properties $(\mathrm{P}_1^{\mathcal{I}})$, $(\mathrm{P}_2^{\mathcal{I}})$ and $(\mathrm{P}_3^{\mathcal{I}})$ are necessary for specifying indifferences, they are not sufficient for ensuring that the indifferences specified are sensible. For example, suppose $i \succ \ldots \succ h \succ \ldots \succ j$ for an arbitrary candidate h by $\mathbf{V}_k$ and that we only state $v_{ji}^{\mathcal{I}} = 1$. Then, the resulting matrix $\mathbf{V}_k + \mathbf{V}_k^{\mathcal{I}}$ would not represent a possible vote; we would have $i \sim j$ but neither $i \sim h$ nor $j \sim h$. We would also be expected to state $v_{hi}^{\mathcal{I}} = 1$ and $v_{jh}^{\mathcal{I}} = 1$ for any candidate h contained between i and j in a ranking encoded by $\mathbf{V}_k$. We require an additional condition to ensure that any specified indifferences in $\mathbf{V}_k^{\mathcal{I}}$ are sensible with respect to the matrix $\mathbf{V}_k$. This is given by the following theorem.

Theorem 3.2. Suppose $\mathbf{V}$ is a pairwise comparison matrix satisfying Theorem 3.1 and $\mathbf{V}^{\mathcal{I}}$ is an indifference matrix satisfying $(\mathrm{P}_1^{\mathcal{I}})$, $(\mathrm{P}_2^{\mathcal{I}})$ and $(\mathrm{P}_3^{\mathcal{I}})$. Consider any pair of distinct candidates i, j with any candidate h such that $i \succ \ldots \succ h \succ \ldots \succ j$ by $\mathbf{V}$. Then $v_{hi}^{\mathcal{I}} = 1$ $(i \sim h)$ and $v_{jh}^{\mathcal{I}} = 1$ $(j \sim h)$ when the following holds: $v_{\alpha\beta}^{\mathcal{I}} = 0 \vee \left((v_{\alpha\kappa})_k + (v_{\alpha\kappa})_k^{\mathcal{I}}\right) = \left((v_{\beta\kappa})_k + (v_{\beta\kappa})_k^{\mathcal{I}}\right)$ for all $\kappa \neq \alpha \neq \beta$.

Proof. Suppose $v_{\alpha\beta}^{\mathcal{I}} = 0 \vee \left((v_{\alpha\kappa})_k + (v_{\alpha\kappa})_k^{\mathcal{I}}\right) = \left((v_{\beta\kappa})_k + (v_{\beta\kappa})_k^{\mathcal{I}}\right)$ holds for all $\kappa \neq \alpha \neq \beta$ when $v_{ij}^{\mathcal{I}} = 1$ for any distinct pair of candidates i, j. First, we prove $j \sim h$. From $\mathbf{V}$, we know the following: $v_{ih} = 1$, $v_{hi} = 0$, $v_{jh} = 0$ and $v_{hj} = 1$. Then, by $(\mathrm{P}_3^{\mathcal{I}})$, we know $v_{ih}^{\mathcal{I}} = v_{hj}^{\mathcal{I}} = 0$. We then have $\left((v_{ih})_k + (v_{ih})_k^{\mathcal{I}}\right) = \left((v_{jh})_k + (v_{jh})_k^{\mathcal{I}}\right) = 1$. Since $v_{jh} = 0$, it must be that $v_{jh}^{\mathcal{I}} = 1$ and hence $j \sim h$.

Next, we prove $i \sim h$. The condition $\left((v_{\alpha\kappa})_k + (v_{\alpha\kappa})_k^{\mathcal{I}}\right) = \left((v_{\beta\kappa})_k + (v_{\beta\kappa})_k^{\mathcal{I}}\right)$ $(\kappa \neq \alpha \neq \beta)$ applies for any distinct pairs of candidates α, β when $v_{\alpha\beta}^{\mathcal{I}} = 1$. Since $v_{jh}^{\mathcal{I}} = 1$, we may then write $\left((v_{h\kappa})_k + (v_{h\kappa})_k^{\mathcal{I}}\right) = \left((v_{j\kappa})_k + (v_{j\kappa})_k^{\mathcal{I}}\right)$ $(\kappa \neq h \neq j)$. From $\mathbf{V}$, we know $v_{ji} = 0$. We then have $\left((v_{hi})_k + (v_{hi})_k^{\mathcal{I}}\right) = \left((v_{ji})_k + (v_{ji})_k^{\mathcal{I}}\right)$. Since $v_{hi} = 0$, it follows that $v_{hi}^{\mathcal{I}} = 1$ and hence $i \sim h$. □

Theorem 3.2 provides a method for verifying all indifferences specified within $\mathbf{V}^{\mathcal{I}}$. The main idea is to directly compare the preferences between two candidates i and j. If $v_{ij}^{\mathcal{I}} = 1$, then we expect i and j to essentially be the same candidate; i and j should each defeat, lose to, or be indifferent to, the same candidates. Therefore, by comparing their vectors of preferences or indifferences, given by $\mathbf{V}_i + \mathbf{V}_i^{\mathcal{I}}$ for i and $\mathbf{V}_j + \mathbf{V}_j^{\mathcal{I}}$ for j, we can determine whether any specified indifferences are sensible. We simply need to verify $v_{\alpha\beta}^{\mathcal{I}} = 0 \vee \left((v_{\alpha\kappa})_k + (v_{\alpha\kappa})_k^{\mathcal{I}}\right) = \left((v_{\beta\kappa})_k + (v_{\beta\kappa})_k^{\mathcal{I}}\right)$ for all α, β, κ such that $\kappa \neq \alpha \neq \beta$. The overall complexity for this is $O(n^3)$, however the complexity for verifying only the strict preferences remains $O(n^2)$. This is still practical, as we demonstrate in Section 5.

We now describe the additional cryptographic operations required to permit indifferences in VERICONDOR. When a vote is cast, VERICONDOR converts it into a pairwise comparison matrix $\mathbf{V}_k$ and an indifference matrix $\mathbf{V}_k^{\mathcal{I}}$. Then, VERICONDOR encrypts both $\mathbf{V}_k$ and $\mathbf{V}_k^{\mathcal{I}}$ using the modified ballot construction equations below.

$$\forall i \in C\colon \left(b_{ii}\right)_k = 1 \wedge \left(b_{ii}^{\mathcal{I}}\right)_k = 1 \tag{$\mathrm{BC}_1^{\mathcal{I}}$}$$

$$\forall i, j \in C, i \neq j : \left(b_{ij}\right)_k = g_0^{\left(x_{ij}\right)_k} g_1^{\left(v_{ij}\right)_k} \wedge \left(b_{ij}^{\mathcal{I}}\right)_k = g_0^{\left(x_{ij}^{\mathcal{I}}\right)_k} g_1^{\left(v_{ij}^{\mathcal{I}}\right)_k} \tag{$\mathrm{BC}_2^{\mathcal{I}}$}$$

$$\forall i, j \in C\colon \left(Y_{ij}\right)_k = g_1^{\left(x_{ij}\right)_k} \wedge \left(Y_{ij}^{\mathcal{I}}\right)_k = g_1^{\left(x_{ij}^{\mathcal{I}}\right)_k} \tag{$\mathrm{BC}_3^{\mathcal{I}}$}$$

Here, $\mathbf{X}_k = \left(x_{ij}\right)_k$ and $\mathbf{X}_k^{\mathcal{I}} = \left(x_{ij}^{\mathcal{I}}\right)_k$ denote the randomness used to encrypt $\mathbf{V}_k$ and $\mathbf{V}_k^{\mathcal{I}}$ respectively. The main difference in the encryption here is the usage of g_1 in $(\mathrm{BC}_2^{\mathcal{I}})$, which is necessary for

utilising of a zero knowledge proof of equality as we will explain. Let $B_k = \langle b_k, Y_k \rangle$ denote the encrypted ballot for $\mathbf{V}_k$ and $B_k^{\mathcal{I}} = \langle b_k^{\mathcal{I}}, Y_k^{\mathcal{I}} \rangle$ denote the encrypted ballot for $\mathbf{V}_k^{\mathcal{I}}$. The following logical relations must be fulfilled.

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

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

The system also maintains the two matrices $\mathbf{S}$ and $\mathbf{T}$ for tallying. The tallying procedure is slightly different to account for the new matrix $\mathbf{V}_k^{\mathcal{I}}$ used in votes. We give the new tallying procedure below.

$$\forall i, j \in C, k \in \mathbb{C} \colon s_{ij} = s_{ij} + \left(x_{ij}\right)_k + \left(x_{ij}^{\mathcal{I}}\right)_k \tag{$\mathrm{T}_1^{\mathcal{I}}$}$$

$$\forall i, j \in C, k \in \mathbb{C} \colon t_{ij} = t_{ij} + \left(v_{ij}\right)_k + \left(v_{ij}^{\mathcal{I}}\right)_k \tag{$\mathrm{T}_2^{\mathcal{I}}$}$$

We utilise the following tally verification equations for voting with indifferences. The procedure to verify these equations remains the same as the case for strict preference; one must simply compute and compare the left-hand side and right-hand side of each equation for equality, and notify the election organisers if any comparison fails.

$$g_0^{s_{ij}} g_1^{t_{ij}} = \prod_{k \in \mathbb{C}} \left(b_{ij}\right)_k \cdot \left(b_{ij}^{\mathcal{I}}\right)_k \tag{$\mathrm{TV}_1^{\mathcal{I}}$}$$

$$g_1^{s_{ij}} = \prod_{k \in \mathbb{C}} \left(Y_{ij}\right)_k \cdot \left(Y_{ij}^{\mathcal{I}}\right)_k \tag{$\mathrm{TV}_2^{\mathcal{I}}$}$$

To verify the well-formedness of encrypted ballots permitting indifferences, we must first verify that $\mathbf{V}_k$ is a correct pairwise comparison matrix of strict preferences satisfying Theorem 3.1, and then verify that $\mathbf{V}_k^{\mathcal{I}}$ is an indifference matrix satisfying Theorem 3.2. The former is straightforward to achieve using the ZKP discussed in Section 3.3 for strict preferences, albeit with a small modification that takes into account the new ballot construction equations (specifically the new usage of g_1 in ($\mathrm{BC}_2^{\mathcal{I}}$)). It is less straightforward to verify that $\mathbf{V}_k^{\mathcal{I}}$ is an indifference matrix satisfying Theorem 3.2. Consider the following equivalence.

$$\left(b_{ij}\right)_k \cdot \left(b_{ij}^{\mathcal{I}}\right)_k = g_0^{\left(x_{ij}\right)_k + \left(x_{ij}^{\mathcal{I}}\right)_k} \cdot g_1^{\left(v_{ij}\right)_k + \left(v_{ij}^{\mathcal{I}}\right)_k}$$

For all $i, j \in [0, n-1]$, the above must hold. As our protocol is additively homomorphic by design, the result contains the summation $\left(v_{i\kappa}\right)_k + \left(v_{i\kappa}^{\mathcal{I}}\right)_k$. Hence, we may simply prove equality in zero knowledge to ensure Theorem 3.2 holds for a pair of distinct candidates i, j in an encrypted ballot. This is done through the following proof of well-formedness for encrypted indifference matrices.

$$P_{WF}^{\mathcal{I}}\left\{ B_k^{\mathcal{I}} = \left\langle b_k^{\mathcal{I}}, Y_k^{\mathcal{I}} \right\rangle \right\} =$$

$$P_K\Bigg\{ \left(x_{ij}^{\mathcal{I}}\right)_k \colon \left(\log_{g_0}\left(b_{ij}^{\mathcal{I}}\right)_k = \log_{g_1}\left(Y_{ij}^{\mathcal{I}}\right)_k\right) \vee \left(\log_{g_0}\left(b_{ij}^{\mathcal{I}}\right)_k / g_1 = \log_{g_1}\left(Y_{ij}^{\mathcal{I}}\right)_k\right) \tag{$\mathrm{C}_1^{\mathcal{I}}$}$$

$$\wedge \left(i \neq j \Rightarrow \left(\log_{g_0}\left(b_{ij}\right)_k = \log_{g_1}\left(Y_{ij}\right)_k\right) \vee \left(\log_{g_0}\left(b_{ij}^{\mathcal{I}}\right)_k = \log_{g_1}\left(Y_{ij}^{\mathcal{I}}\right)_k\right)\right) \tag{$\mathrm{C}_2^{\mathcal{I}}$}$$

$$\wedge \Bigg(\log_{g_0}\left(b_{ij}^{\mathcal{I}}\right)_k = \log_{g_1}\left(Y_{ij}^{\mathcal{I}}\right)_k \tag{$\mathrm{C}_3^{\mathcal{I}}$}$$

$$\vee \forall \kappa \neq i \neq j \colon \left(\log_{g_0}\left(\left(b_{i\kappa}\right)_k \cdot \left(b_{i\kappa}^{\mathcal{I}}\right)_k\right) = \log_{g_0}\left(\left(b_{j\kappa}\right)_k \cdot \left(b_{j\kappa}^{\mathcal{I}}\right)_k\right)\right)$$

$$\vee \left(\log_{g_0}\left(\left(b_{i\kappa}\right)_k \cdot \left(b_{i\kappa}^{\mathcal{I}}\right)_k\right) = \log_{g_0}\left(\left(b_{j\kappa}\right)_k \cdot \left(b_{j\kappa}^{\mathcal{I}}\right)_k\right)\right) \quad \Bigg) \Bigg\}$$

The conditions (C_1^I) and (C_2^I) ensure that $\mathbf{V}_k^I$ is an indifference matrix at every entry other than those on the diagonal (we do not need a ZKP for the diagonal entries as these have a fixed value of 1). The other condition (C_3^I) ensures that $\mathbf{V}_k^I$ is a well-formed indifference matrix through application of Theorem 3.2. Almost all conditions in this proof have $O(n^2)$ complexity, with only (C_3^I) having a complexity of $O(n^3)$. We discuss the time complexity of this proof further in Section 5.

3.6 Electing a Winner

In a Condorcet election, several candidates may end up with a tie (forming a Condorcet cycle). To break the tie, several Condorcet methods have been proposed in the past literature, such as Black's method [30], the Minimax method [31, 32], the Schulze method [3], Copeland's method [33], Ranked Pairs [34], Dodgson's method [35] and the Kemeny-Young method [36, 37]. 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 [38] 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 satisfy different voting system criteria and hence anyone running an election using VERICONDOR can choose a suitable and efficient Condorcet method for their election.

Of the methods we consider, the Schulze method, Copeland's method and the Ranked Pairs method are straightforward to utilise with rankings that contain indifference; these methods can be directly used with VERICONDOR. Permitting indifference across Black's method and the Minimax method is less straightforward. We explain the difficulties or interpretations for both of these methods as we consider them.

Black's method. Black's method uses the Borda count system in the event that there is no Condorcet winner for an election [30]. Adapting VERICONDOR to support Black's method is straightforward for strict preferences: we simply run VERICONDOR and DRE-Borda [18] 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.

It is also possible to permit indifferences within Black's method, although this is not ideal and may lead to strategic voting [39]. Black's method is still compatible with VERICONDOR, assuming that voters are not strategic when constructing their votes; a requirement which is difficult to achieve in practice.

The Minimax method. The Minimax method elects the winner whose greatest pairwise defeat is smaller than the greatest pairwise defeat of any other candidate [31, 32]. Electing a winner using the Minimax method in VERICONDOR is straightforward for strict preferences since the final sum matrix contains all information needed; the winner is the result of $\arg\min_{i \in C}(\max_{j \in C}(t_{ji}))$.

Three interpretations are possible in order to utilise the Minimax method with votes containing indifferences, namely: *pairwise opposition*, *winning votes* and *margins* [40]. Of these three interpretations, only the latter two satisfy Condorcet's criterion; as such, we only focus on these two interpretations. Let d_{ij} denote the number of voters ranking i above j in $\mathbf{T}$. Under *winning votes*, $score_{ij}$ is defined as the number of voters ranking i above j only when this score exceeds the number ranking j above i, i.e. $score_{ij} = d_{ij}$ if $d_{ij} > d_{ji}$ and $score_{ij} = 0$ otherwise. Under *margins*, $score_{ij}$ is defined as the margin of victory of i over j, i.e. $score_{ij} = d_{ij} - d_{ji}$. Using either definition

of $score$, a winner for the Minimax method may then be determined using the standard procedure, albeit substituting $score_{ji}$ for t_{ji}. In either case, the Minimax method is simple and efficient; 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 π where $\pi_0 = i$ and $\pi_{n-1} = j$. We then define the *strength* of a path π as $s_\pi = \min_{l \in C}(\mathbf{T}(\pi_l, \pi_{l+1}) - \mathbf{T}(\pi_{l+1}, \pi_l))$. 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., $w_{ij} = \max_{\pi \in P_{ij}}(s_\pi)$. 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 [41]. We may model this precisely using a results matrix; in a pairwise comparison between a candidate i and a candidate j, let $\mathbf{\Gamma} = (\gamma_{ij})$ and define $\gamma_{ij} = 1$ if $t_{ij} > t_{ji}$, $\gamma_{ij} = \frac{1}{2}$ if $t_{ij} = t_{ji}$ and $\gamma_{ij} = 0$ if $t_{ij} < t_{ji}$. 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 [42]. 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 $\left\{1, \frac{1}{2}, 0\right\}$. 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 $t_{i_0 j_0} > t_{i_1 j_1}$ [34]. In the event where $t_{i_0 j_0} = t_{i_1 j_1}$, 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) [43]. If $t_{i_0 j_0} = t_{i_1 j_1}$ 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 [34]; 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* [34]: 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 [34]; 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 $O(n^2)$.

Summary. 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.

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

Table 2. Summary of support for different Condorcet methods in VERICONDOR.

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.

4 SECURITY ANALYSIS

We next prove the E2E-verifiability of VERICONDOR and the secrecy of ballots 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 $m - \eta$ honest votes (assuming m confirmed votes) based on the DDH assumption [44]. Since we require a secure hash function for transforming an interactive ZKP to a non-interactive ZKP based on Fiat-Shamir heuristics, our proofs are in a random oracle model. In this analysis, we will not explicitly consider the ZKPs in our security proofs for simplicity. We note that ZKPs serve

to prove the well-formedness of an encrypted ballot and reveal nothing more than the truth of the statement. First of all, we state the DDH assumption below.

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 $\Omega = R$.

4.1 E2E-Verifiability

We show that VERICONDOR satisfies the three requirements of end-to-end verifiability, namely the *cast as intended*, *recorded as cast* and *tallied as recorded* requirements [9]. It is straightforward to see that our system satisfies the "cast as intended" requirement based on the well-established voter-initiated auditing technique [24]. 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 selection), 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 $\mathbf{X}_k$. A voter may choose to audit their vote for any number of times. 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. We utilise a similar proof technique as the one used for DRE-ip [17]. For our proof, we consider the case for strict preferences within Theorem 4.1. Theorem 4.1 shows that if the ballots are well-formed, i.e., $P_{WF}\{B_k\}$ holds, and the tally verification equations (TV_1) and (TV_2) 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 $\forall i, j \in C\colon \prod_{k\in\mathbb{C}}(b_{ij})_k = g_0^{s_{ij}} g_0^{t_{ij}} \wedge \prod_{k\in\mathbb{C}}(Y_{ij})_k = g_1^{s_{ij}}$ 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 $\forall i, j \in C\colon \prod_{k\in\mathbb{C}}(Y_{ij})_k = g_1^{s_{ij}}$ holds. We show that $\prod_{k\in\mathbb{C}}(b_{ij})_k = g_0^{s_{ij}} g_0^{t_{ij}}$ holds if and only if $t_{ij} = \sum_{k\in\mathbb{C}}(v_{ij})_k$ also holds for all $i, j \in C$.

($\Rightarrow$) Suppose $\prod_{k\in\mathbb{C}}(b_{ij})_k = g_0^{s_{ij}} g_0^{t_{ij}}$ for all $i, j \in C$. By definition of $(b_{ij})_k$, we have $(b_{ij})_k = g_0^{(x_{ij})_k} g_0^{(v_{ij})_k}$ and hence $\prod_{k\in\mathbb{C}}(b_{ij})_k = \prod_{k\in\mathbb{C}} g_0^{(x_{ij})_k} g_0^{(v_{ij})_k} = g_0^{\sum_{k\in\mathbb{C}}(x_{ij})_k} g_0^{\sum_{k\in\mathbb{C}}(v_{ij})_k}$. By applying ($\mathrm{TV}_2$), we have $s_{ij} = \sum_{k\in\mathbb{C}}(x_{ij})_k$. It is then clear that $t_{ij} = \sum_{k\in\mathbb{C}}(v_{ij})_k$.

($\Leftarrow$) Suppose $t_{ij} = \sum_{k\in\mathbb{C}}(v_{ij})_k$ for all $i, j \in C$. By definition of $(b_{ij})_k$, we have $(b_{ij})_k = g_0^{(x_{ij})_k} g_0^{(v_{ij})_k}$ and also $\prod_{k\in\mathbb{C}}(b_{ij})_k = g_0^{\sum_{k\in\mathbb{C}}(x_{ij})_k} g_0^{\sum_{k\in\mathbb{C}}(v_{ij})_k}$. By applying ($\mathrm{TV}_2$) and our initial assumption we get the result $\prod_{k\in\mathbb{C}}(b_{ij})_k = g_0^{s_{ij}} g_0^{t_{ij}}$. This completes the proof. □

Theorem 4.1 holds regardless of whether votes are represented as full matrices or triangular matrices. If indifference is permitted, one must substitute $(v_{ij})_k$ for $(v_{ij})_k + (v_{ij}^{\mathcal{I}})_k$. One can then show, albeit with $P_{WF}^{\mathcal{I}}$, ($\mathrm{TV}_1^{\mathcal{I}}$) and ($\mathrm{TV}_2^{\mathcal{I}}$), that $\mathbf{T}$ is the correct tally when an indifference matrix is also included in the construction of $\mathbf{S}$ and $\mathbf{T}$.

4.2 Ballot Secrecy

We now consider the notion of *ballot secrecy* for an election, 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 [45] and define privacy to be maintained if an attacker colluding with a set of η

voters has a negligible chance to distinguish between any two elections where both elections have the same partial tally of honest votes. In 2015, Bernhard et al. [46] proposed a newer game-based definition of privacy called BPRIV. The game model used in the BPRIV definition assumes "an honest single trustee" who not only performs verifiable decryption but also is tasked to remove duplicate ballots. The authors remark that the BPRIV definition can be extended in a multi-trustee scenario. However, we find the BPRIV definition not suitable in our case since VERICONDOR does not involve any tallying authorities (or trustees). We note that the need to remove duplicate ballots in the BPRIV model is motivated by addressing a replay attack reported on Helios 2.0 [28]. The replay attack is possible in Helios 2.0 because there is no unique identity included in the ballot ZKP (this is not a simple omission, but an inherent issue in the protocol design as no such identity is available at the time when the ZKP is generated). However, the replay attack is prevented in VERICONDOR by design since each ballot is identified by a unique index, which is included in the ballot well-formedness ZKP (as part of the input to the hash function).

Assumption 4.2. 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, Ω_d)

In the experiment, the challenger first randomly selects two elements g, and A, from the group $\mathbb{G}_q$. 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 $DH_g(A, B)$. O 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 in the experiment.

The advantage of an adversary $\mathcal{A}$, against $Exp_{\mathcal{A}}^{RND}(\lambda)$ 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 $\mathcal{A}$, $Adv_{\mathcal{A}}^{RND}(\lambda) \leq negl(\lambda)$.

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

Proof. This Lemma is proved as Lemma 4 in Kurosawa and Nojima [47]. □

Assumption 4.3. Let us consider the following security experiment $Exp_{\mathcal{A}}^{RND1}(\lambda)$. In this experiment, the challenger samples g, and A, randomly from $\mathbb{G}_q$. 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 samples an element $B \in_R \mathbb{G}_q$. The oracle O selects one of them depending upon a bit d chosen by the challenger in the experiment $Exp_{\mathcal{A}}^{RND1}(\lambda)$, and computes Ω_d as shown in the description. Then O returns B, and Ω_d to the adversary.

$Exp_{\mathcal{A}}^{RND1}(\lambda)$
$g \xleftarrow{\$} \mathbb{G}_q$
$A \xleftarrow{\$} \mathbb{G}_q$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(g, A)$
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, Ω_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 $\mathcal{A}$, $Adv_{\mathcal{A}}^{RND1}(\lambda) \leq negl(\lambda)$.

Lemma 4.2. *Assumption 4.2 implies 4.3.*

Proof. The lemma can be easily proven by application of the triangle inequality for computational indistinguishability [48]: for any three distributions D_0, D_1 and D_2, we have that $SD(D_0, D_2) \leq SD(D_0, D_1) + SD(D_1, D_2)$, where SD denotes the statistical difference between two distributions. Using the triangle inequality, one can show that $Adv_{\mathcal{A}}^{RND1}(\lambda) \leq 2 * Adv_{\mathcal{A}}^{RND}(\lambda)$. □

We shall now define the indistinguishability notion in a Condorcet election. The following security experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ models the scenario where the adversary chooses two different sets of votes having the same cardinality and the same tally. The challenger randomly picks one of them and converts the set of votes into encrypted ballots following our Condorcet e-voting scheme. The adversary's task is to identify the set of votes that was selected by the challenger.

Definition 4.1. Consider the security experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$. 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 are initialized to empty. It also stores three $(n-1) \times (n-1)$ triangular matrices X, V_0, and V_1 that are initialized to $\mathbf{0}_{n,n}$. That is, $X(i, j) = 0$, and $V_0(i, j) = V_1(i, j) = 0, \forall i, j \in [0, n-1], i < j$. Assume that Γ is the set of all $(n-1) \times (n-1)$ triangular comparison matrices. Hence, for each $\tau \in \Gamma$, $\tau(i, j) \in \{0, 1\}, \forall i, j \in [0, n-1], i < j$. The challenger then invokes $\mathcal{A}_0$. $\mathcal{A}_0$ is given access to the oracle O. $\mathcal{A}$ passes two inputs $v_0 \in \Gamma$, and $v_1 \in \Gamma$ to the oracle O, every time it is called. O selects random $(n-1) \times (n-1)$ triangular matrix $x \in \mathbb{Z}_p^{n(n-1)/2}$, and generates c_0, and c_1 as shown in the experiment. They represent the ballots for the two sets of votes, v_0 and v_1. It stores c_0, and c_1 in BB_0, and BB_1. It stores the cumulative values of the selected randomnesses in the triangular matrix 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 $V_0 = V_1$. Thus, in this experiment the adversary is required to distinguish between the bulletin boards of two Condorcet elections in which the votes are chosen by the adversary with the only condition that the tallies V_0, and V_1 must be equal. If the tallies are not equal, then the adversary can trivially distinguish between both bulletin boards. The two tallies in both bulletin boards must therefore be the same, however the individual inputs may be different and chosen by the adversary themselves.

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

$O(v_0, v_1)$
$x \xleftarrow{\$} \mathbb{Z}_p^{n(n-1)/2}$
$c_0(i,j) \leftarrow (g_0^{x(i,j)} g_0^{v_0(i,j)}, g_1^{x(i,j)}) : i,j \in [0,n-1], i < j$
$c_1(i,j) \leftarrow (g_0^{x(i,j)} g_0^{v_1(i,j)}, g_1^{x(i,j)}) : i,j \in [0,n-1], i < j$
$X(i,j) \leftarrow X(i,j) + x(i,j) : i,j \in [0,n-1], i < j$
$BB_i \leftarrow BB_i \bigcup \{c_i\} : i = 0,1$
$V_0(i,j) \leftarrow V_0(i,j) + v_0(i,j) : i,j \in [0,n-1], i < j$
$V_1(i,j) \leftarrow V_1(i,j) + v_1(i,j) : i,j \in [0,n-1], i < j$

The advantage of an adversary $\mathcal{A}$, against $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ 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 $Adv_{\mathcal{A}}^{IND-Vote}(\lambda) \leq negl(\lambda)$.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B}$, against the security experiment $Exp_{\mathcal{B}}^{RND1}(\lambda)$ of Assumption 4.3. $\mathcal{B}$ works as follows; it receives as input g_0, and A. It invokes $\mathcal{A}$ with (g_0, A). Whenever $\mathcal{A}_0$ makes a query to the oracle O with input v_0, and v_1, $\mathcal{B}$ also makes $n(n-1)/2$ queries of the form $(v_{0t}(i,j), v_{1t}(i,j))$ in the experiment $Exp_{\mathcal{B}}^{RND1}(\lambda)$ for all $i,j \in [0,n-1], i < j$, and $t \in [0,\eta-1]$. Here, η is the total number of queries to O made by $\mathcal{A}_0$ in $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$. $\mathcal{B}$ does not know the value of η. Therefore, 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 $(B(i,j), \Omega(i,j))$ as the value returned by the oracle in $Exp_{\mathcal{B}}^{RND1}(\lambda)$.

For each of the first $\eta - 1$ queries, $\mathcal{B}$ has to make $n(n-1)/2$ queries to its internal oracle, and receives $n(n-1)/2$ responses which we denote as $(B_k(i,j), \Omega_k(i,j))$, for $i,j \in [0,n-1], i < j, k \in [0,\eta-2]$. The last query made by $\mathcal{A}_0$ in $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ is $(v_{0\eta-1}, v_{1\eta-1})$. Let $V(i,j) = \sum_{k=0}^{\eta-1} v_{0k}(i,j) = \sum_{k=0}^{\eta-1} v_{1k}(i,j), \forall i,j \in [0,n-1], i < j$. Now, $\mathcal{B}$ selects random $X(i,j) \xleftarrow{\$} \mathbb{Z}_p$ for all $i,j \in [0,n-1], i < j$. $\mathcal{B}$ then sets the following values for all $i,j \in [0,n-1], i < j$:

$$B_{\eta-1}(i,j) = \frac{A^{X(i,j)}}{\prod_{k=0}^{\eta-2} B_k(i,j)}$$

$$\Omega_{\eta-1}(i,j) = \frac{g^{X(i,j)+V(i,j)}}{\prod_{k=0}^{\eta-2} \Omega_k(i,j)}$$

Let us denote $BB_d = \{(B_k, \Omega_k) : k \in [0,\eta-1]\}$. All zero knowledge proofs are simulated by $\mathcal{B}$. Invoke $\mathcal{A}_1$ with BB_d. If $\mathcal{A}_1$ can identify d, so can $\mathcal{B}$. If $\mathcal{B}$ can make a correct guess of the value of η, the advantage of $\mathcal{B}$ will be same as that of $\mathcal{A}$. If $\mathcal{B}$ cannot guess it correctly, then the advantage will be 0. Therefore, the following holds.

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

One may then re-arrange the above to produce $Adv_{\mathcal{B}}^{RND1}(\lambda) \geq 1/poly(\lambda) * Adv_{\mathcal{A}}^{IND-Vote}(\lambda)$, then it is the case that $Adv_{\mathcal{A}}^{IND-Vote}(\lambda) \leq poly(\lambda) * Adv_{\mathcal{B}}^{RND1}(\lambda)$. □

Now, we show that our scheme is secure against active adversary that can corrupt some but not all the voters. We show using the experiment $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ to prove this fact. The difference between $Exp_{\mathcal{A}}^{IND-Vote}(\lambda)$ and a real Condorcet election is that in a real election, the attacker may compromise some but not all the voters. We show that an attacker that corrupts an arbitrary number of voters can only learn the partial tally of honest voters. Hence, the attacker can only infer whatever the partial tally allows them to infer. The attacker cannot distinguish between all possible voting patterns that results into the same tally. In order for proving the fact, we consider an election where some voters are honest and others are corrupt. The attacker chooses the votes of corrupt voters, whereas the honest votes are selected by the challenger from one of two sets of votes that correspond to the same partial tally. The task of the attacker is to identify the correct set of votes that represent the set of honest votes. All the NIZK proofs of well-formedness are simulated by the challenger. So, the attacker will have negligible advantage to distinguish the NIZK proofs from real ones. So, the actual advantage of the attacker comes from the encrypted ballots. The following lemma proves that the attacker will have negligible advantage in distinguishing between the two sets of honest votes.

Lemma 4.4. *Let us assume that in an arbitrary Condorcet election, there are m voters, where $m \in poly(\lambda)$. The voters are indexed as $P_i : i \in [1, m]$. Let H be the set of indices of honest users. All other voters are compromised by the attacker $\mathcal{A}$. We denote by V_i, the triangular comparison matrix that represents the vote of P_i, for $i \in [1, m]$. Consider two sets of triangular comparison matrices $\Psi_0 = \{\overrightarrow{V}_{0i} : i \in [1, |H|]\}$ and $\Psi_1 = \{\overrightarrow{V}_{1i} : i \in [1, |H|]\}$, satisfying $\sum_{i=1}^{|H|} \overrightarrow{V}_{0i} = \sum_{i=1}^{|H|} \overrightarrow{V}_{1i}$. As such, the adversary will not be able to distinguish between $\{V_i : i \in H\} = \Psi_0$ and $\{V_i : i \in H\} = \Psi_1$.*

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. $\mathcal{B}_0$ queries O with (v, v) whenever $\mathcal{A}$ enters a vote v for a corrupt voter. The two parameters passed to O are the same. For the uncompromised voters in the set H, $\mathcal{B}_0$ selects $\overrightarrow{V}_{0i}$, and $\overrightarrow{V}_{1i}$, for some $i \in [1, |H|]$. $\mathcal{B}_0$ sends $(\overrightarrow{V}_{0i}, \overrightarrow{V}_{1i})$ to O. Since, the number of honest voters is $|H|$ which is the same as $|\Psi_0|$ and $|\Psi_1|$, $\mathcal{B}_0$ can set a unique pair of votes from (Ψ_0, Ψ_1) that will represent the vote of the i^{th} honest voter. Once all m queries to O have been done, $\mathcal{B}_0$ returns. Then $\mathcal{B}_1$ is invoked with BB_d ($d \in \{0, 1\}$). $\mathcal{B}$ sends it to $\mathcal{A}$. If $\mathcal{A}$ can identify d, so can $\mathcal{B}_1$. Hence, the result holds. □

Using the same method, one can show that the Condorcet e-voting scheme using indifference is also secure against an adversary that can corrupt an arbitrary number of voters, and the adversary only learns the partial tally of honest votes from the bulletin board.

5 PERFORMANCE ANALYSIS

We build a proof of concept implementation of VERICONDOR in Java[4], supporting two different ranking methods, based on *strict preference* and *indifference* respectively. We evaluate the system performance both theoretically and empirically using microbenchmarks. The theoretical analysis is found matching the empirical results.

5.1 Theoretical Estimates

To estimate the theoretical performance of our system for ranking under *strict preference* and *indifference*, we mainly focus upon the number of exponentiations required in each case. This is because exponentiation is the most demanding operation in VERICONDOR, regardless of which

[4]The source code of our implementation: https://osf.io/n2afp/?view_only=bd54965fd92640fdbf02547d0dd23a34.

ranking method is chosen. We consider the performance of generating individual ballots, verifying the well-formedness proofs, and lastly verifying the tally verification equations.

When generating individual ballots, we must perform their ballot construction equations as well as the equations for constructing their well-formedness proofs. Summing the exponentiations for these steps gives us an estimate for the running time of ballot creation. To estimate the running time of well-formedness proof verification, we may simply enumerate the exponentiations required for the verification procedures. The running time of tally verification is slightly more complex to estimate as it is defined over a set of confirmed ballots $\mathbb{C}$, hence we also consider the number of multiplications required for tally verification (in terms of $|\mathbb{C}|$) in addition to the number of exponentiations required. Our estimates are summarised in Table 3. For comparison, we also give the performance results of applying a full matrix to represent a ballot under *strict preference* based on Theorem 3.1, as done in our earlier paper [19]. Our improved representation, based on Corollary 3.1, uses only a triangular matrix, and reduces the computation by about half.

Protocol	**Ballot Construction**	**Well-formedness Verification**	**Tally Verification**	
	Exponentiations	**Exponentiations**	**Exponentiations**	**Multiplications**
Strict Preference (Full Matrix)	$16n^2 - 5n$	$19n^2 - 5n$	$3n^2$	$n^2(2\|\mathbb{C}\| + 1)$
Strict Preference (Triangular Matrix)	$\frac{15}{2}n^2 - \frac{5}{2}n$	$\frac{17}{2}n^2 - \frac{5}{2}n$	$\frac{3}{2}n^2 - \frac{3}{2}n$	$\left(n^2 - n\right)\left(\|\mathbb{C}\| - \frac{1}{2}\right)$
Indifference (Full Matrix)	$11n^3 - 9n^2 + 16n$	$12n^3 - 3n^2 + 18n$	$3n^2$	$n^2(2\|\mathbb{C}\| + 3)$

Table 3. Performance estimates for the different protocols in VERICONDOR.

Our estimates show that VERICONDOR remains $O(n^2)$ in terms of the number of exponentiations required when constructing ballots containing strict preferences only or verifying their well-formedness proofs. The overall complexity when using a triangular matrix (Corollary 3.1) is the same as that of using the full matrix (Theorem 3.1), however the factors are roughly halved. We show that this results in a more desirable performance in practice in Section 5.2. When indifferences are permitted, the number of exponentiations required rises to $O(n^3)$ for both ballot construction and proof verification. This complexity is less desirable, although still remains practical. It is up to the election organiser to decide whether to permit voters to state indifferences between candidates at the cost of lengthening the ballot construction and proof verification procedures. Regardless of the protocol chosen however, the tally verification procedures all require $O(n^2)$ exponentiations. The number of multiplications needed for tally verification is a product of both n^2 and $|\mathbb{C}|$, which is likely to be insignificant in practice as the running time of VERICONDOR is dominated by the more expensive exponentiation operations.

5.2 Microbenchmarks

We recorded the average time to run ballot creation, well-formedness verification and tally verification of all three protocols discussed previously within our proof of concept implementation. We consider two cases for each construction: 1) a 2048-bit p, g_0 and g_1 with a 224-bit q for 112-bit security; and 2) a 3072-bit p, g_0 and g_1 with a 256-bit q for 128-bit security. To maintain consistency

with the benchmarking procedure in our earlier paper [19], we make use of the same number of votes (50 votes [19]) when benchmarking tally verification (for a complete implementation, it would be desirable to consider larger numbers of votes, as well as the storage needed for these). Our results are shown across Figures 6, 7 and 8.

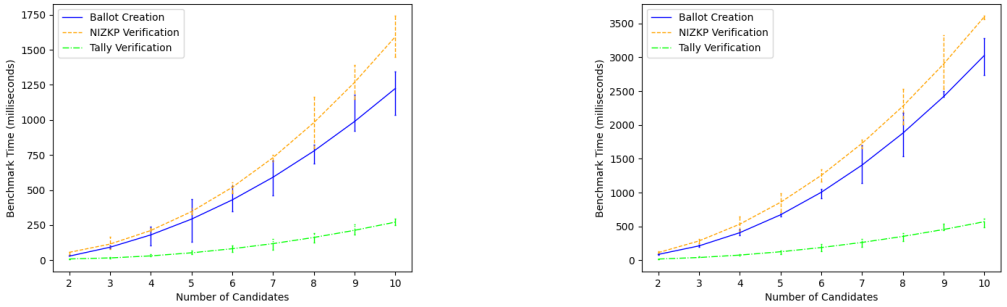

Fig. 6. 112-bit (left) and 128-bit (right) security benchmarks for VERICONDOR utilising full matrix.

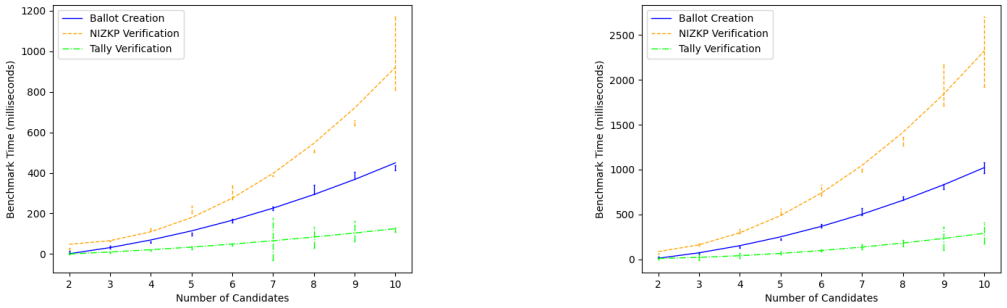

Fig. 7. 112-bit (left) and 128-bit (right) security benchmarks for VERICONDOR utilising triangular matrix.

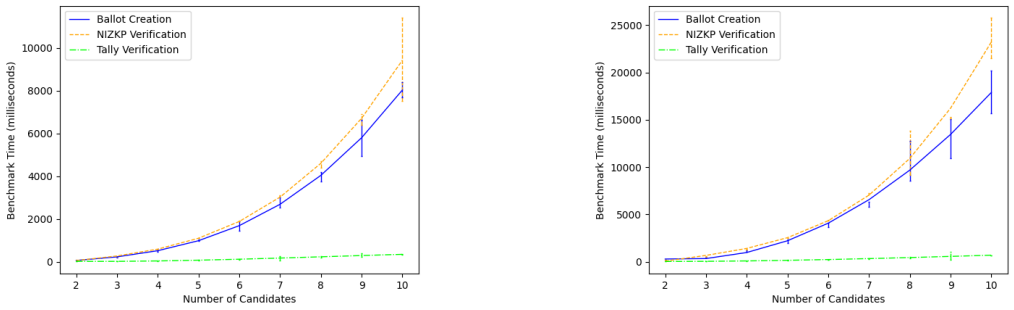

Fig. 8. 112-bit (left) and 128-bit (right) security benchmarks for VERICONDOR permitting indifference.

Our microbenchmark results remain consistent with the theoretical runtime analysis of each construction. Tally verification for all constructions is the least demanding procedure, followed by ballot construction, then well-formedness verification. It is clear from Figures 6, 7 and 8 that exponentiation has the largest effect on the average running times of VERICONDOR, regardless of the chosen ranking method. We also expect the usage of a triangular matrix instead of a full one for *strict preference* to approximately halve the time needed to perform the ballot construction, well-formedness verification and tally verification procedures; it is clear from Figures 6 and 7 that this expectation holds in our experiment.

We may see from Figure 8 that permitting indifferences within votes has a significant impact on the running time of ballot construction and proof verification. This is due to the need for two $n \times n$ matrices to represent each vote as well as the need for more complex proofs compared to the case of *strict preference*. Reducing the average running time for ballot construction and well-formedness verification would be beneficial here. This may be achieved in practice by making use of Elliptic Curve Cryptography (ECC) instead (the specification of the protocol would remain unchanged). Additionally, each entry in a ballot may be generated and verified in parallel. It should be noted that these improvements can also be applied to the ranking method based on *strict preference*, or for any elections that contain a significantly large number of candidates. However, the latter kind of election is typically uncommon in practice. Huber et al. note that Condorcet elections typically contain fewer than a dozen candidates [49] due to the need for voters to rank every candidate against all others in a Condorcet election. For a large number of candidates, this ranking can quickly become tedious.

6 RELATED WORK

A number of verifiable e-voting schemes have been proposed in the literature. The vast majority of these schemes focus on plurality voting [10–13, 50]. Few schemes are designed for ranked-choice voting. Examples include DRE-Borda [18] for Borda count, the work of Ramchen et al. for Instant Runoff Voting (IRV) [51], the work of Lee and Doi for Condorcet voting [52] and the works of Hertel et al. [7], Cortier et al. [8] and Huber et al. [49] for various voting systems that include Condorcet voting. For comparisons with VERICONDOR, we consider only those systems that provide a verifiable solution to Condorcet voting.

Lee and Doi proposed in 2005 a TA-based *partially* verifiable voting scheme designed specifically to elect a Condorcet winner for an election [52]. 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 [24] 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 trustworthy authorities to perform the mixing and decryption operations. Lee and Doi specify the requirement of two different authorities: a tallying authority for performing the encryption and permutation procedures and a "judging authority" for performing decryption and determining the Condorcet winner. Lee and Doi's scheme also assumes that candidates are ranked in strict preference only, with no support for indifference between candidates. The computational cost of their scheme is $O(n^3)$, while it is $O(n^2)$ for strict preferences in VERICONDOR.

Hertel et al. proposed in 2021 several new instantiations to the Ordinos [53] voting system to permit verifiable Condorcet voting [7]. Ordinos uses an additively homomorphic t-out-of-n threshold public key encryption scheme where n is the number of secret key shares with $t \leq n$ of these shares being necessary for successful decryption. It requires a set of TAs to perform homomorphic aggregation of votes and computation the tally through a secure multi-party computation

(MPC) protocol. Ordinos aims to maintain a *tally-hiding* property, whereby only the final winner of an election is revealed and the remainder of the tally remains hidden (except to the TAs). The new instantiations for Ordinos provided by Hertel et al. also utilise a comparison matrix scheme for verifiable Condorcet voting, with well-formedness maintained through NIZKPs. The scheme proposed by Hertel et al. assumes candidates are ranked in strict preference only, with no support for indifference within rankings. The cost of generating each ballot including the well-formedness proof in their scheme is $O(n^3)$, whilst it is $O(n^2)$ for strict preferences in VERICONDOR.

Cortier et al. provide a collection of MPC building blocks for designing tally-hiding e-voting schemes and demonstrate how a Condorcet e-voting system is constructed using these building blocks [8]. The authors motivate the usage of an ElGamal cryptosystem over Pallier and require a set of TAs to perform aggregation of votes. The authors utilise a shuffling scheme to verify whether comparison matrices are well-formed. This scheme, first presented by Haines et al. [54], is based on the observation that shuffling the rows and columns of a comparison matrix according to a ranked list preserves the well-formedness property of the original matrix. Cortier et al. extend this scheme to permit indifference by allowing voters to send a vector of encrypted bits that specifies which candidates are indifferent within their vote. The voter must then modify the initial matrix into a transformed matrix that encodes their provided indifference. The overall cost of ballot construction and proof verification in their scheme is $O(n^2)$, which does improve upon the $O(n^3)$ costs in VERICONDOR for indifferences. However, Cortier et al. require TAs to verify the shuffle. The number of exponentiations required by the TAs depends on the information leakage deemed acceptable by the election organisers and is either $49.5mn^2a \log n$ (partial MPC) or $49.5mn^2a \log n + 198n^3a \log m$ (full MPC), assuming m voters, n candidates and a authorities [8]. As VERICONDOR is free from any TAs, it avoids all the associated computational or communication costs between them.

Huber et al. describe *Kryvos*, a tally-hiding verifiable e-voting system that supports various voting methods including Condorcet. The authors utilise Pedersen commitments [55] and require a set of talliers and a voting authority. The talliers must carry out homomorphic aggregation of votes, with a designated tallier computing both the election result and a Succinct Non-interactive Argument of Knowledge (SNARK) [56] proving knowledge of the tally [49]. Kryvos utilises comparison matrices for Condorcet voting but does not permit voters to be indifferent between candidates. The tally is computed by aggregating all matrices as Pedersen commitments, with the designated tallier publishing the Condorcet winner or set of potential winners. Generating a ballot together with its well-formedness proof is $O(n^3)$ in Kryvos for strict preferences, while it is $O(n^2)$ in VERICONDOR.

Summary. VERICONDOR provides multiple benefits not present in the reviewed works. The most notable is no requirement for any TA, which is a necessity for other works [7, 8, 49, 52]. The removal of TAs simplifies election management; a voter can fully verify the tallying integrity by themselves without involving any TA. Additionally, VERICONDOR achieves the $O(n^2)$ complexity for the computation of each ballot for strict preferences, which is probably the best one can hope for given the use of a $n \times n$ comparison matrix to record a Condorcet vote. VERICONDOR is also flexible enough to support indifferences within votes, albeit with a less desirable time complexity of $O(n^3)$. This complexity still remains practical however. At the end of an election, VERICONDOR presents an aggregated comparison matrix, from which a Condorcet winner or an alternative winner based on the Condorcet methods can be announced. This method avoids revealing the individual ballots, hence preventing the so-called Italian attack [49]. Based on definitions by Huber et al. [49], our protocol can be considered as "partially tally-hiding". A "fully tally-hiding" scheme only reveals who is the winner without publishing any tally. This property can be achieved by involving TAs as shown in several works [7, 8, 49]. We note that while the tally is hidden from voters, it can still be computed by the TAs. Hence, the TAs need to be trusted to keep the tally

secret. Since VERICONDOR does not involve any TAs, it does not provide the "fully tally-hiding" property. Extending VERICONDOR to support this property is a subject for further research.

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 novel methods to prove well-formedness of the matrix with exceptional efficiency. It supports ranking candidates in strict preference, as well as permitting indifference. The computational costs are exceptionally efficient for strict preferences at $O(n^2)$ for n candidates, and remain practical for indifferences at $O(n^3)$. 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 event that a Condorcet winner does not exist. It protects the secrecy of ballots and limits any colluding set of voters to learn nothing more than the total tally and the partial tally of their votes. When the voting machine is completely breached by a powerful adversary, the tallying integrity remains intact due to E2E-verifiability, and information leakage is limited to only the partial tally at the time of compromise.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. The first author is funded by an EPSRC studentship (No. 2436418). The last author is supported by an EPSRC grant (EP/T014784/1). We thank Julian Liedtke for confirming the $O(n^3)$ complexity for the ballot well-formedness proof in their paper [7].

REFERENCES

[1] A. Sen. Majority Decision and Condorcet Winners. *SC&W*, 54(2):211–217, 2020.
[2] H.P. Young. Condorcet's Theory of Voting. *APSR*, pages 1231–1244, 1988.
[3] M. Schulze. The Schulze Method of Voting. *arXiv preprint arXiv:1804.02973*, 2018.
[4] R. Winger. Ballot Access News. http://ballot-access.org/2021/03/28/march-2021-ballot-access-news-print-edition/, 2021. Online; accessed 29 November 2022.
[5] Libertarian Party of Washington. Constitution of the Libertarian Party of Washington State. https://lpwa.org/wp-content/uploads/2022/07/LPWA_Constitution_updated_at_convention_26March2022.pdf#page=10, 2022. Online; accessed 29 November 2022.
[6] W.H. Riker. Voting and the Summation of Preferences: An Interpretive Bibliographical Review of Selected Developments during the Last Decade. *APSR*, 55(4):900–911, 1961.
[7] F. Hertel, N. Huber, J. Kittelberger, R. Küsters, J. Liedtke, and D. Rausch. Extending the Tally-Hiding Ordinos System: Implementations for Borda, Hare-Niemeyer, Condorcet, and Instant-Runoff Voting. *Cryptology ePrint Archive*, 2021.
[8] V. Cortier, P. Gaudry, and Q. Yang. A Toolbox for Verifiable Tally-Hiding E-Voting Systems. In *ESORICS*, pages 631–652. Springer, 2022.
[9] F. Hao and P.YA. Ryan. *Real-World Electronic Voting: Design, Analysis and Deployment.* CRC Press, 2016.
[10] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R.L. Rivest, P.YA. Ryan, E. Shen, and A.T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. *EVT*, 8:1–13, 2008.
[11] P.YA. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt à Voter: A Voter-Verifiable Voting System. *IEEE TIFS*, 4(4):662–673, 2009.
[12] A. Kiayias, T. Zacharias, and B. Zhang. DEMOS-2: Scalable E2E Verifiable Elections without Random Oracles. In *ACMCCS*, pages 352–363, 2015.
[13] B. Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.
[14] B. Adida, O.De Marneffe, O. Pereira, JJ. Quisquater, et al. Electing a University President using Open-Audit Voting: Analysis of Real-World Use of Helios. *EVT/WOTE*, 9(10), 2009.
[15] F. Hao, M.N. Kreeger, B. Randell, D. Clarke, S.F. Shahandashti, and P.HJ. Lee. Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting. In *EVT/WOTE 2014*, 2014.
[16] F. Hao. DRE-i and Self-Enforcing E-Voting. *Chapter of Real World Electronic Voting*, page 343, 2016.

[17] S.F. Shahandashti and F. Hao. DRE-ip: A Verifiable E-Voting Scheme without Tallying Authorities. In *ESORICS*, pages 223–240. Springer, 2016.

[18] S. Bag, M.A. Azad, and F. Hao. E2E Verifiable Borda Count Voting System without Tallying Authorities. In *ARS*, pages 1–9, 2019.

[19] L. Harrison, S. Bag, H. Luo, and F. Hao. VERICONDOR: End-to-End Verifiable Condorcet Voting without Tallying Authorities. In *ASIACCS 2022*, pages 1113–1125, 2022.

[20] F. Hao, S. Wang, S. Bag, R. Procter, S.F. Shahandashti, M. Mehrnezhad, E. Toreini, R. Metere, and L.YJ. Liu. End-to-End Verifiable E-Voting Trial for Polling Station Voting. *IEEE S&P*, 18(6):6–13, 2020.

[21] C. Culnane and S. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *IEEE CSF 2014*, pages 169–183. IEEE, 2014.

[22] A. Kiayias, A. Kuldmaa, H. Lipmaa, J. Siim, and T. Zacharias. On the Security Properties of E-Voting Bulletin Boards. In *SCN 2018*, pages 505–523. Springer, 2018.

[23] L. Hirschi, L. Schmid, and D. Basin. Fixing the Achilles Heel of E-Voting: The Bulletin Board. In *IEEE CSF 2021*, pages 1–17. IEEE, 2021.

[24] J. Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. *EVT*, 7:14–14, 2007.

[25] J. Camenisch and M. Stadler. Proof Systems for General Statements about Discrete Logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

[26] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *IEEE S&P 2018*, pages 315–334. IEEE, 2018.

[27] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *TACT*, pages 186–194. Springer, 1986.

[28] V. Cortier and B. Smyth. Attacking and Fixing Helios: An Analysis of Ballot Secrecy. *JCS*, 21(1):89–148, 2013.

[29] N.J.A. Sloane. The Online Encyclopedia of Integer Sequences. http://oeis.org, 2022. Sequence A000670.

[30] D. Black et al. The Theory of Committees and Elections. 1958.

[31] P.B. Simpson. On Defining Areas of Voter Choice: Professor Tullock on Stable Voting. *QJoE*, 83(3):478–490, 1969.

[32] G.H. Kramer. A Dynamical Model of Political Equilibrium. *JET*, 16(2):310–334, 1977.

[33] A.H. Copeland. A Reasonable Social Welfare Function. Technical report, mimeo, 1951. University of Michigan, 1951.

[34] T.N. Tideman. Independence of Clones as a Criterion for Voting Rules. *SC&W*, 4(3):185–206, 1987.

[35] C. Dodgson. A Method of Taking Votes on More Than Two Issues. *TOCAE*, 1876.

[36] J.G. Kemeny. Mathematics Without Numbers. *Daedalus*, 88(4):577–591, 1959.

[37] H.P. Young and A. Levenglick. A Consistent Extension of Condorcet's Election Principle. *JOAM*, 35(2):285–300, 1978.

[38] J. Bartholdi, C.A. Tovey, and M.A. Trick. Voting Schemes for Which It Can Be Difficult to Tell Who Won The Election. *SC&W*, 6(2):157–165, 1989.

[39] Iain McLean Arnold B Urken et al. *Classics of Social Choice*. University of Michigan Press, 1995.

[40] R.B. Darlington. Are Condorcet and Minimax Voting Systems the Best? *arXiv preprint arXiv:1807.01366*, 2018.

[41] D.G. Saari and V.R. Merlin. The Copeland Method: I.: Relationships and the Dictionary. *ET*, 8(1):51–76, 1996. ISSN 09382259, 14320479. URL http://www.jstor.org/stable/25054952.

[42] D.D. Nguyen. Using Social Choice Function Vs. Social Welfare Function To Aggregate Individual Preferences In Group Decision Support Systems. *IJMIS*, 18(3):167–172, 2014.

[43] T.M. Zavist and T.N. Tideman. Complete Independence of Clones in the Ranked Pairs Rule. *SC&W*, 6(2):167–173, 1989.

[44] D.R. Stinson and M. Paterson. *Cryptography: Theory and Practice*. CRC press, 2018.

[45] J.D.C. Benaloh. Verifiable Secret-Ballot Elections. 1989.

[46] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions. In *IEEE S&P 2015*, pages 499–516. IEEE, 2015.

[47] K.Kurosawa and R.Nojima. Simple adaptive oblivious transfer without random oracle. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, pages 334–346. Springer, 2009. ISBN 978-3-642-10366-7.

[48] J.K.H. Iv, M. Georgiou, A.J. Malozemoff, and T. Shrimpton. Security Foundations for Application-Based Covert Communication Channels. In *IEEE S&P 2022*, pages 1971–1986. IEEE, 2022.

[49] N. Huber, R. Kuesters, T. Krips, J. Liedtke, J. Mueller, D. Rausch, P. Reisert, and A. Vogt. Kryvos: Publicly Tally-Hiding Verifiable E-Voting. *Cryptology ePrint Archive*, 2022.

[50] S. Bell, J. Benaloh, M.D. Byrne, D. DeBeauvoir, B. Eakin, P. Kortum, N. McBurnett, O. Pereira, P.B. Stark, D.S. Wallach, et al. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. In *EVT/WOTE 2013*, 2013.

[51] K. Ramchen, C. Culnane, O. Pereira, and V. Teague. Universally Verifiable MPC and IRV Ballot Counting. In *ICOFCADS*, pages 301–319. Springer, 2019.

[52] YC. Lee and H. Doi. On the Security of Condorcet Electronic Voting Scheme. In *International Conference on Computational and Information Science*, pages 33–42. Springer, 2005.

[53] R. Küsters, J. Liedtke, J.Müller, D. Rausch, and A. Vogt. Ordinos: A Verifiable Tally-Hiding E-Voting System. In *EuroS&P 2020*, pages 216–235. IEEE, 2020.
[54] T. Haines, D. Pattinson, and M. Tiwari. Verifiable Homomorphic Tallying for the Schulze Vote Counting Scheme. In *WCOVS:TTE*, pages 36–53. Springer, 2019.
[55] T.P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *AICC*, pages 129–140. Springer, 1991.
[56] J. Groth. On the Size of Pairing-Based Non-Interactive Arguments. In *TAAOCT*, pages 305–326. Springer, 2016.

APPENDIX

A PROOF OF THEOREM 3.1

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 $\rho = (c_0, c_1, \ldots, c_{n-1})$ 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 ρ one at a time, starting with c_0. Element c_0 is preferred to $c_1, c_2, \ldots, c_{n-1}$ by ρ. 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 $\mathbf{V}$ as $\mathbf{V}^\Sigma$. 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 $\rho = (i_0, i_1, \ldots, i_{n-1})$. This is one of the $n!$ possible votes for this election. This completes the proof. □

B NIZKP FOR COROLLARY 3.1

From Corollary 3.1, we know that a triangular matrix $\mathbf{U}$ represents one of the $n!$ votes for an n-candidate election iff the vector $\left(\left[\sum_{i=c+1}^{n-1}(u_{ci})_k\right] + c - \left[\sum_{i=0}^{c-1}(u_{ic})_k\right]\right)_{c=0}^{n-1}$ is a permutation of C. For the latter, consider the following equivalence.

$$g_0^c \cdot \frac{\prod_{i=c+1}^{n-1} b_k(c,i)}{\prod_{i=0}^{c-1} b_k(i,c)} = g_0^{\sum_{i=c+1}^{n-1}(x_{ci})_k - \sum_{i=0}^{c-1}(x_{ic})_k} \cdot g_0^{\left[\sum_{i=c+1}^{n-1}(u_{ci})_k\right] + c - \left[\sum_{i=0}^{c-1}(u_{ic})_k\right]}$$

It is then sufficient to prove the following (based on Bag et al. [18]).

$$\bigwedge_{c' \in C} c' \in \left(\left[\sum_{i=c+1}^{n-1} (u_{ci})_k \right] + c - \left[\sum_{i=0}^{c-1} (u_{ic})_k \right] \right)_{c=0}^{n-1}$$

These relations are equivalent to the following logical statement.

$$\bigvee_{c \in C} g_0^c \cdot \frac{\prod_{i=c+1}^{n-1} (b_{ci})_k}{\prod_{i=0}^{c-1} (b_{ic})_k} = g_0^{\sum_{i=c+1}^{n-1} (x_{ci})_k - \sum_{i=0}^{c-1} (x_{ic})_k} \cdot g_0^{\mathcal{J}}$$

For all $\mathcal{J} \in C$, exactly one of the above disjunctions should hold. Suppose that, for a fixed $\mathcal{J}$, the m^{th} disjunction is true. Denote $(b_m^{\Pi-})_k = \frac{\prod_{i=m+1}^{n-1} (b_{mi})_k}{\prod_{i=0}^{m-1} (b_{im})_k}$ and $(\mathbf{X}_m^{\Sigma-})_k = \sum_{i=m+1}^{n-1} (x_{mi})_k - \sum_{i=0}^{m-1} (x_{im})_k$. We then have $g_0^m \cdot (b_m^{\Pi-})_k = g_0^{(\mathbf{X}_m^{\Sigma-})_k} g_0^{\mathcal{J}}$. 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:

$$\forall i \in C - \{m\} : t'_i = g_0^{r_i} \left(\frac{g_0^i \cdot (b_i^{\Pi-})_k}{g_0^{\mathcal{J}}} \right)^{c_i} \pmod{p}$$

Let the grand challenge be $c = H(k, j, b_k^{\Pi-}, t'_0, t'_1, \ldots, t'_{n-1})$, where $H(\cdot)$ denotes a cryptographic hash function. The prover then computes:

$$c_m = c - \sum_{i \in C - \{m\}} c_i \pmod{q}$$

$$r_m = u_m - c_m (\mathbf{X}_m^{\Sigma-})_k \pmod{q}$$

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

$$\forall i \in C : g_0^{r_i} \equiv \frac{t'_i}{\left(\frac{g_0^i \cdot (b_i^{\Pi-})_k}{g_0^{\mathcal{J}}} \right)^{c_i}} \pmod{p}$$