Versatile Prêt à Voter: Handling multiple election methods with a unified interface

Zhe Xia[1], Chris Culnane[1], James Heather[1], Hugo Jonker[2], Peter Y. A. Ryan[2], Steve Schneider[1], and Sriramkrishnan Srinivasan[1]

[1] Department of Computing, University of Surrey, Guildford GU2 7XH, U.K.
`{z.xia,c.culnane,j.heather,s.schneider,s.srinivasan}@surrey.ac.uk`
[2] Faculté des Sciences, de la Technologie et de la Communication,
University of Luxembourg, L-1359 Luxembourg
`{hugo.jonker,peter.ryan}@uni.lu`

Abstract. A number of end-to-end verifiable voting schemes have been introduced recently. These schemes aim to allow voters to verify that their votes have contributed in the way they intended to the tally and in addition allow anyone to verify that the tally has been generated correctly. These goals must be achieved while maintaining voter privacy and providing receipt-freeness. However, most of these end-to-end voting schemes are only designed to handle a single election method and the voter interface varies greatly between different schemes. In this paper, we introduce a scheme which handles many of the popular election methods that are currently used around the world. Our scheme not only ensures privacy, receipt-freeness and end-to-end verifiability, but also keeps the voter interface simple and consistent between various election methods.

Keywords: Prêt à Voter, voting scheme, end-to-end verifiability, receipt-freeness, simple and consistent voter interface

1 Introduction

In a traditional secret ballot election, voters mark their choice on a piece of paper and drop it into a box. The ballots are mixed together to break the link between the voter and her choice. Then these ballots are tallied under scrutiny. While the secret ballot meets its desired goals of ensuring voter privacy, lack of transparency and verifiability is considered a problem. There is no way for the voter to verify that her vote has contributed correctly to the tally and significant trust must be placed in the election officials to have carried out the election procedures and tally correctly.

End-to-end verifiable voting schemes aim to address these issues. These schemes allow voters to verify that their votes have contributed in the way they intended to the tally (individual verifiability) and in addition allow anyone to verify that the tally has been generated correctly (universal verifiability). These goals must be achieved while maintaining voter privacy. In addition, the voter should not be able to prove to others how she voted as adversaries can then coerce or bribe the voter to vote in a certain way (receipt-freeness).

1.1 Motivation

The most common election method is the First-Past-The-Post (FPTP) method, where a voter simply puts a mark next to the candidate of her choice. The majority of the existing end-to-end voting schemes are designed to handle FPTP elections. Some examples are Prêt à Voter (PaV) [9], [27], Scratch & Vote [2], Punchscan [24], Scantegrity [8], [7], Bingo Voting [6] and MarkPledge [18], [1].

However, many other election methods exist and are used widely. Typically, these allow the voter to indicate multiple preferences or allot a full or partial ranking of the candidates. A plethora of tallying methods also exist and are used around the world. In the Appendix, we provide a summary of the election methods we will discuss in this paper.

It is argued in the decision theory community that these ranked elections are superior as less votes are wasted and that they offer resistance to strategic voting. However, they introduce potential coercion problems. For example, if the election consists of a large number of candidates, a very large number of possible candidate rankings exist. Adversaries can force voters to cast their votes using specific orderings, and check whether ballots with these unique orderings appear among the cast ballots. This has been referred to as the Italian attack in the media and literature. We discuss Italian attacks further in Section 5.

Ranked election methods are typically less discussed in the end-to-end voting literature. There are a few notable exceptions. Several schemes [15], [31], [21] have been designed for Instant Runoff Voting (IRV) elections. Shuffle-Sum [5] handles both IRV and the Single Transferable Vote (STV), but it does not directly handle partial rankings. Condorcet elections can be handled in the scheme introduced in [10]. However, in order to foil the Italian attack, its user interface is quite different: instead of ranking the candidates, every voter is required to cast a number of ballots, where each ballot is a pairwise comparison of two candidates.

No generic end-to-end verifiable voting scheme exists that can handle a wide variety of election methods. This is an important consideration as several different elections, employing different election methods are often held at the same time. For example, on election day, a voter in the polling station may need to cast several ballots, each for a separate election using a different election method. The average voter is unable to understand complicated instructions and procedures to cast their vote. Therefore, the voting interface needs to be kept simple and consistent to avoid confusion. Two attempts, one with cryptography and one without cryptography, in this direction have been made in [32] and [25], where a generic solution to handle various election methods is introduced. Unfortunately, neither of them is receipt-free in ranked elections.

1.2 Our contribution

In this paper, we gather together many diverse concepts and building blocks in the literature, unifying them into one generic scheme which handles a number of popular election methods. In addition to achieving the goals of voter privacy, receipt-freeness and end-to-end verifiability, our scheme has a simple and consistent voter interface across various election methods. In order to enable our scheme to be used in high-profile political elections, we also aim to achieve robustness so that the election can be run even in the presence of a minority of dishonest election officials and is able to recover from cheating when it is detected. We summarize our contributions in more specific detail in Section 6.

1.3 Structure of the paper

In Section 2 we summarize the various cryptographic building blocks we will employ in our scheme. This is followed by an overview of the proposed scheme in Section 3. As discussed, the user interface is kept consistent for different election methods. The exact details of how the votes are processed for the different election methods is abstracted from the voter and we describe these details in Section 4. We then provide an informal analysis of the security properties of the scheme in Section 5 before concluding in Section 6.

2 Building Blocks

2.1 Paillier cipher

The Paillier cipher [20] is an efficient, semantically secure public key cryptosystem which provides the additive homomorphic property. It is a fundamental building block of our scheme and works as follows: let n be an RSA modulus $n = pq$, where p and q are large primes. Let g be an integer of order a multiple of n modulo n^2, e.g. $g = n + 1$. The public key is (g, n), and the secret key is $\lambda = lcm((p-1),(q-1))$. To encrypt $m \in Z_n$, we randomly choose $r \in Z_n^*$ and compute the ciphertext $c = E_{pk}(m, r) = g^m r^n \pmod{n^2}$. To decrypt c, we compute $m = L(c^\lambda \bmod n^2)/L(g^\lambda \bmod n^2) \bmod n$, where the L-function takes input values from the set $S_n = \{u < n^2 | u = 1 \bmod n\}$ and $L(u) = (u-1)/n$.

- *Homomorphic property:* For two ciphertexts under the same public key $c_1 = E_{pk}(m_1, r_1)$ and $c_2 = E_{pk}(m_2, r_2)$, we have $E_{pk}(m_1, r_1) \cdot E_{pk}(m_2, r_2) = E_{pk}(m_1 + m_2, r_1 \cdot r_2)$. Moreover, for a value $k \in Z_n$, we have $E_{pk}(m, r)^k = E_{pk}(k \cdot m, r^k)$. This property is fundamental to the construction of our scheme and will be used extensively along with the Baudron homomorphic counter [3] detailed in the following section.
- *Paillier re-encryption:* Given a Paillier ciphertext $c = g^m r^n \pmod{n^2}$, a re-encryption of this ciphertext can be generated without knowledge of the secret key λ. A value $t \in Z_n^*$ is randomly selected and the re-encryption c' of c is calculated as $c' = c \cdot t^n = g^m (t \cdot r)^n \pmod{n^2}$.

- *Threshold Paillier:* The key pair for the Paillier cipher can be jointly generated by threshold parties, so that each party has a share of the secret key, but no single party knows the entire secret key. This technique can be found in [14], [12]. Moreover, ciphertexts can be decrypted by these threshold parties in a verifiable manner [13].
- *Verifiable shuffle for Paillier:* In a verifiable shuffle, a mix server receives a batch of ciphertexts, re-encrypts each ciphertext and then outputs the results in a random order. It also publishes a proof so that the correctness of the shuffle can be publicly verified, but the proof should not reveal the relationship between the inputs and the outputs. Verifiable shuffle for Paillier can be designed using existing techniques from [16], [4], [19], [22].

2.2 Baudron's homomorphic counter

Suppose there are $k+1$ candidates and the total number of eligible voters is M, where $M < 2^L$. We can define a set of counters $\{2^0, 2^L, 2^{2L}, \ldots, 2^{kL}\}$ as the election parameters, one for each candidate. Encryptions corresponding to each counter represent votes for the candidate who has been assigned the counter. Multiplying these encrypted votes together results in an encrypted counter where the received votes for each candidate is kept in a separate area of the counter without overflow between adjacent areas of the counter. For more technical details, please see [3].

2.3 Plaintext Equivalence Test (PET)

Suppose (g, n) is the Paillier public key and the secret key λ is shared among threshold parties. For two ciphertexts $c_1 = E_{pk}(m_1, r_1) = g^{m_1} {r_1}^n \pmod{n^2}$ and $c_2 = E_{pk}(m_2, r_2) = g^{m_2} {r_2}^n \pmod{n^2}$, the Plaintext Equivalent Test (PET) [30] can be used to check whether these two ciphertexts contain the same plaintext, without revealing either plaintext. Denote $c = c_1/c_2 = g^{m_1 - m_2}(r_1/r_2)^n$ $\pmod{n^2}$. If $m_1 = m_2$, for some random value $r \in Z_n$, both c and c^r will contain the 0 plaintext. But if $m_1 \neq m_2$, c^r will contain a random plaintext. Therefore, if c^r is threshold decrypted, the result can tell whether the two ciphertexts contain the same plaintext without revealing either plaintext.

2.4 Binary Conversion and Plaintext Inequivalence Test (PIT)

For a Paillier ciphertext $c = E_{pk}(m, r)$, where the binary representation of its L-bit long plaintext is $m = m_1 m_2 \cdots m_L$, the Binary Conversion technique [28] can be used to convert the ciphertext c into separate encryptions of every individual bit of the plaintext m. This process can be illustrated as

$$E_{pk}(m_1 m_2 \cdots m_L, r) \rightarrow E_{pk}(m_1, r_1), E_{pk}(m_2, r_2), \ldots, E_{pk}(m_L, r_L)$$

Binary Conversion needs to be carried out by the threshold parties, and it is publicly verifiable. The above process can be reversed by anyone as:

$$E_{pk}(m, r') = \prod_{i=1}^{L} E_{pk}(m_{L+1-i}, r_{L+1-i})^{2^{i-1}}$$

Moreover, for any two Paillier ciphertexts which have been converted into the encryption of individual bits of the plaintext, the threshold parties can apply techniques in [11] to check whether these two ciphertexts contain the same plaintext, or which ciphertext contains larger plaintext. This phase is also publicly verifiable, and does not reveal either plaintext.

3 System overview

We first describe the ballot generation phase in our proposed scheme. We then describe the vote capture phase, which is where the individual voter interacts with the system. This is the phase that is consistent from the voter's point of view, no matter what election method is being employed. The techniques employed in the vote processing phase are different depending on which election method is being used and the details are abstracted from the voter. We will describe the various vote processing techniques in Section 4.

- **Ballot generation:** All ballot forms are generated by a trusted party before the election. We trust this party for privacy, but the integrity of the election result does not rely on this party. Suppose there are 5 candidates in a sample election: Alice, Bravo, Charlie, Delta and Echo. They are assigned counters 2^0, 2^L, 2^{2L}, 2^{3L} and 2^{4L} respectively, where 2^L is larger than the total number of eligible voters in the election. These are published prior to the election as system parameters. A ballot, as shown in Figure 1, consists of two columns with a vertical perforation down the middle. The left hand column lists the candidate names in a random order. The barcode in the right hand column contains an encrypted value for each candidate and a proof. The encrypted values are called "onions" for historical reasons, and their ordering should match the candidate list. The proof proves that each onion encrypts a unique counter. To generate the proof, the party first generates a list of pseudo ciphertexts:

  $$\{E_{pk}(2^0, 1), E_{pk}(2^L, 1), E_{pk}(2^{2L}, 1), E_{pk}(2^{3L}, 1), E_{pk}(2^{4L}, 1)\}$$

  Anyone can check that this list is well-formed because all the randomisations are 1. For each ballot, the party generates a proof that the onion list is a shuffle of the pseudo ciphertexts.
- **Vote capture:** To prevent the local officials from learning the candidate ordering, the ballots are delivered to the polling stations in sealed envelopes and these sealed ballots are handed to the voters during the voting phase. The voter opens the envelope to obtain the ballot and can then randomly

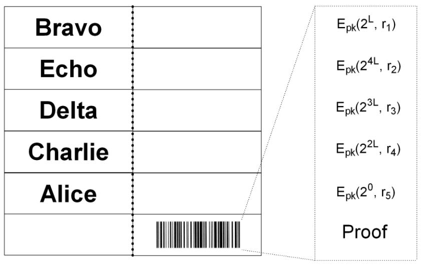

Fig. 1. A ballot form example

decide whether to audit the ballot or use it to cast a vote. If the voter chooses to audit her ballot, she submits it to the local officials without marking her choice. After the onions have been threshold decrypted, anyone can check whether the candidate list can be reconstructed from the onions. Once a ballot is audited, it cannot be used to cast a vote, and the voter will be provided with another ballot. The voter can audit several ballots until she is satisfied. Then, the voter marks the ballot as instructed: selecting a single candidate, multiple candidates, or by providing a ranking of candidates. This is pictured in Figure 2.

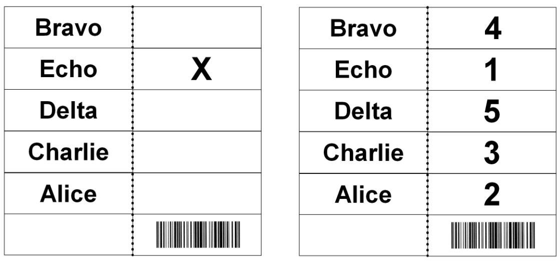

Fig. 2. Completing the ballot form. (a) Single cross. (b) Preference list

As follows, the voter separates the ballot along the perforation and shreds the left hand column. It is important to ensure via some process that the left hand column is destroyed before the voter is allowed to submit the vote. Otherwise, the voter can prove to an adversary how she voted. The remaining right hand column, as shown in Figure 3, contains the vote to be cast. The voter now brings it to the election officials and it is scanned into the election system and digitally signed. The voter retains the signed right hand side as her receipt. All the received votes are published on the bulletin board and the voter can check whether her vote has been recorded correctly by the election system. If not, the signed receipt can be used to initiate a complaint. The use of the Prêt à Voter style ballot form provides two advantages. Firstly, it is simple and very similar to the current paper based ballots that voters are already familiar with. Secondly, if the voter mistakenly casts an invalid vote, e.g. over-vote or under-vote, it can be discovered by the local officials before the vote is scanned and the voter can be assisted in casting a valid vote by being instructed appropriately and being provided with new ballots. Note that the election official cannot violate the vote secrecy by simply looking at the filled in right hand side, as the corresponding left hand side has been destroyed.

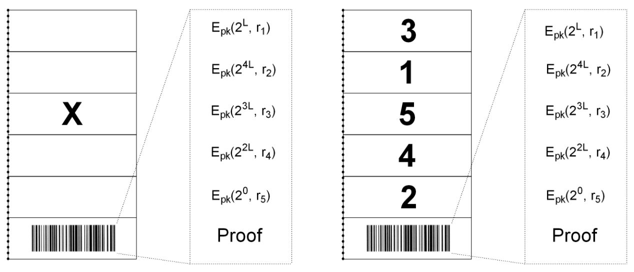

Fig. 3. The vote. (a) Single cross. (b) Preference list

4 Vote Processing

As discussed earlier, the vote processing phase is transparent to voters. At a high level, this phase can be regarded as an oracle. If it was provided with the received votes and was told which election method is used, it will generate the election result based on that election method. Moreover, the oracle will output some data onto the bulletin board. The data is enough to publicly verify the tally, but it provides no information that can be used by an adversary to coerce voters.

4.1 Vote processing for FPTP

The vote processing for FPTP elections is the same as in the Scratch & Vote scheme [2]. When the votes, as shown in Figure 3(a) are received, the proofs for each vote are checked and votes with invalid proofs are discarded. The proof is important to prevent malicious parties from casting invalid vote, e.g. negative vote(s) or multiple votes. For the remaining votes, the onion corresponding to the selected candidate is aggregated into a counter as described previously. Finally, this counter is threshold decrypted and the election result is announced along with the tally for each candidate.

In [31], a strategy is introduced to announce only the election winner without revealing any other information. Firstly, onions are aggregated into the counter. However, instead of decrypting the counter, the threshold parties can apply the Binary Conversion technique to convert it into separate encryptions of each bit of the plaintext. For our sample election, the plaintext is $5L$ bits long, and every L-bit block represents the received vote for a different candidate. The candidate with the most votes can be identified using the *plaintext inequivalence test* (PIT) and this can be publicly verified. The exact number of votes received by each candidate is kept secret in this method.

4.2 Vote processing for Approval Voting

In Approval Voting, a voter can mark one or several crosses, and all the crosses are equally weighted. The vote processing for Approval Voting is similar to FPTP. First, all proofs are checked and votes with invalid proof are removed. All onions corresponding to selected candidates in a ballot are then aggregated into a single ciphertext. From this point on the techniques and options are similar to those described above.

4.3 Vote processing for Supplementary Vote

In Supplementary Vote elections, the votes, as shown in Figure 4, contain either one or two preferences. To tally the received votes, the proof of every vote is checked and all votes with invalid proofs are removed. The onions corresponding to the valid ballots are ordered according to the indicated preference. For example, the votes in Figure 4 will be ordered as $\{\theta_D\}$ and $\{\theta_B, \theta_A\}$ respectively (note that these θ values are ciphertexts and the subscripts are used for notational convenience). Now, the first onion in every vote is selected and aggregated into a counter. The threshold parties then apply the Binary Conversion technique to convert this ciphertext into a number of ciphertexts, where each encrypts the received votes for a particular candidate. After generating a pseudo ciphertext[3] $E_{pk}(Q, 1)$, where Q is the winning quota, the threshold parties apply the PIT to check whether some candidate has received more votes than the quota. If yes, the election ends and this candidate wins. Otherwise, the threshold parties identify the two candidates with the most votes using the PIT, and all other candidates are eliminated.

Suppose *Bravo* and *Delta* are the two remaining candidates. We first generate two pseudo ciphertexts $\bar{\theta}_B = E_{pk}(2^L, 1)$ and $\bar{\theta}_D = E_{pk}(2^{3L}, 1)$ for them respectively. Then we shuffle all the votes using the verifiable shuffle. In the next step, for every vote in the outputs, the threshold parties apply the PET to compare

[3] Note that in the rest of this paper, if we mention pseudo ciphertext, we mean that the ciphertext is generated using the randomisation value 1. Therefore, anyone can verify that the pseudo ciphertext is well-formed.

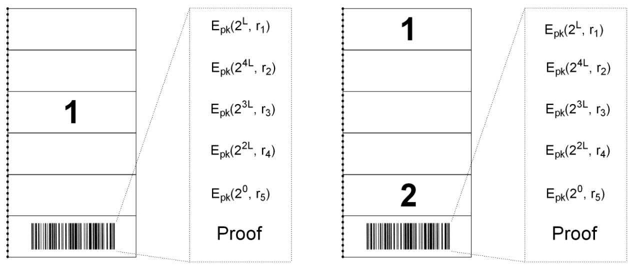

Fig. 4. Supplementary Vote. (a) One preference. (b) Two preferences

its first onion with $\bar{\theta}_B$ and $\bar{\theta}_D$. If it matches with one of the pseudo ciphertexts, this vote will be sorted into the pile for that candidate. For these votes, the second preference will never be used. So if any vote has a second onion, it will be removed from the vote, e.g. θ_A will be removed from the vote $\{\theta_B, \theta_A\}$. For the other votes where the first onion does not match with any of the pseudo ciphertexts, we check whether it contains a second preference. If no, the vote will be removed from the tally. Otherwise, its first onion will be removed, leaving only the second onion, and we leave these votes in an unsorted pile. We now check the two piles of votes for the remaining candidates, if their difference is larger than the number of votes in the unsorted pile, the election ends and the remaining candidate with more votes wins. But if their difference is smaller than the number of votes in the unsorted pile, we aggregate all votes in the three piles into one ciphertext[4]. Then this ciphertext is threshold decrypted and one of the two remaining candidates with the most votes wins.

4.4 Vote processing for Instance Runoff Voting (IRV)

To tally the received votes in IRV elections, proofs are checked and any vote with an invalid proof is discarded. Once again, the onions corresponding to the valid ballots are ordered according to the indicated preference e.g. the vote in Figure 3(b) will be re-written as $\{\theta_E, \theta_A, \theta_B, \theta_C, \theta_D\}$. The first onion of every vote is selected and aggregated into a counter. The threshold parties then apply the Binary Conversion technique to transfer this encrypted counter into a number of ciphertexts, where each encrypts the received votes for a candidate. After generating a pseudo ciphertext of the quota as $E_{pk}(Q, 1)$, the threshold parties can use the PIT to check whether some candidate has received more votes than the quota. If yes, the election ends, and this candidate wins. Otherwise, the threshold parties use the PIT to identify the candidate with the least votes and this candidate is eliminated. Suppose Alice is eliminated in the first round, a pseudo ciphertext for Alice is generated as $\bar{\theta}_A = E_{pk}(2^0, 1)$. Then all votes are inserted into the verifiable shuffle. In the outputs, the threshold parties apply

[4] Note that at this moment, all votes in the three piles contain only one onion.

the PET to locate the onion θ_A in every vote, and this value will be removed from the vote. Again, the first onion of every vote is selected and aggregated into a counter and then the threshold parties apply the Binary Conversion and PIT to check whether some candidate has received more votes than the quota. The above process is repeated until some candidate receives more votes than the quota or only one candidate remains. For a vote with partial rankings, if all preferences are removed, it is removed from the tally.

4.5 Vote processing for Single Transferable Vote (STV)

In STV elections with fraction transfer, if some candidate receives more votes than the quota but not all seats are filled, the votes that exceed the quota need to be transferred. However, the transfer value is not an integer but a fraction. For example, if the quota is q and if a candidate receives m votes, where $m > q$, the transfer value will be $(m - q)/q$. Although this value can be treated as an integer if we multiply all the vote values by q, the Baudron counter can no more be used because there might be overflow between adjacent locations within the counter.

Here, the Shuffle-Sum techniques [5] to tally STV elections can be employed, but with modifications to allow partial ranking. Before the election, two pseudo ciphertexts $\bar{\theta}_0 = E_{pk}(0, 1)$ and $\bar{\theta}_1 = E_{pk}(1, 1)$ are generated. These two values are used to record which are the voter's genuine choices and which are appended choices. Every received vote will first be transferred into the so called *Preference-order table*, e.g. the vote in Figure 4(b) will be transferred as:

Candidate	θ_B	θ_A	θ_E	θ_D	θ_C
Preference	1	2	3	4	5
True/Fake	$\bar{\theta}_1$	$\bar{\theta}_1$	$\bar{\theta}_0$	$\bar{\theta}_0$	$\bar{\theta}_0$
Weight		$E_{pk}(w_v)$			

In the above table, the order of the voter's genuine preferences need to match with the vote, but the appended preferences can be in any order. Anyone can verify that the above table is correctly generated. The following procedures are similar to the Shuffle-Sum protocol: the above table can be transferred between the *First-preference table* and the *Candidate-elimination table* to elect some candidate or to eliminate some candidate from the vote. Note that in any case if all the genuine choices are eliminated, the vote will be discarded. This can be checked that in the third row of the *Preference-order table*, the encrypted value under the first preference encodes plaintext 0.

We will not go into the details of the Shuffle-Sum protocol. For more information, please see [5]. But superior than the Shuffle-Sum protocol, our scheme always enable us to check whether all seats are filled in the first round using the homomorphic tallying. The process is the same as in Supplementary Vote and IRV elections. Therefore, we only need to apply the mixnets tallying if all seats are not filled in the first round.

4.6 Vote processing for Condorcet Voting

In Condorcet Voting, the Binary Conversion technique or PET/PIT are not used, and the proof in the vote does not need to be checked. Instead, each received vote is interpreted as follows: e.g. the ciphertexts in the vote in Figure 3(b) will be arranged as per the preferences as $\{\theta_E, \theta_A, \theta_B, \theta_C, \theta_D\}$. A group of ciphertext triples are then generated.

$$\{\theta_E, \theta_A, \bar\theta_1\}\ \{\theta_E, \theta_B, \bar\theta_1\}\ \{\theta_E, \theta_C, \bar\theta_1\}\ \{\theta_E, \theta_D, \bar\theta_1\}\ \{\theta_A, \theta_B, \bar\theta_1\}$$
$$\{\theta_A, \theta_C, \bar\theta_1\}\ \{\theta_A, \theta_D, \bar\theta_1\}\ \{\theta_B, \theta_C, \bar\theta_1\}\ \{\theta_B, \theta_D, \bar\theta_1\}\ \{\theta_C, \theta_D, \bar\theta_1\}$$

In the above group, for each ciphertext triple, the first two ciphertexts are taken from the vote with the same order, and the third one is a pseudo encryption of plaintext 1, $\bar\theta_1 = E_{pk}(1,1) = g^1 \cdot 1^n \pmod{n^2}$.

Similarly, another group of ciphertext triples are generated, in which the first two ciphertexts are taken from the vote in the reverse order, and the third value[5] is a pseudo encryption of -1, $\bar\theta_{-1} = E_{pk}(-1,1) = g^{-1} \cdot 1^n \pmod{n^2}$.

$$\{\theta_D, \theta_C, \bar\theta_{-1}\}\ \{\theta_D, \theta_B, \bar\theta_{-1}\}\ \{\theta_D, \theta_A, \bar\theta_{-1}\}\ \{\theta_D, \theta_E, \bar\theta_{-1}\}\ \{\theta_C, \theta_B, \bar\theta_{-1}\}$$
$$\{\theta_C, \theta_A, \bar\theta_{-1}\}\ \{\theta_C, \theta_E, \bar\theta_{-1}\}\ \{\theta_B, \theta_A, \bar\theta_{-1}\}\ \{\theta_B, \theta_E, \bar\theta_{-1}\}\ \{\theta_A, \theta_E, \bar\theta_{-1}\}$$

Now, we treat all the ciphertext triples in the above two groups as inputs and insert them into the verifiable shuffle. Note that in the outputs, the last value of the ciphertext triple is no longer a pseudo ciphertext i.e. its randomisation is no longer 1. Then, for each of the output ciphertext triples, the threshold parties decrypt the first two values. Now, in exactly half of the result triples, the two candidates are in the alphabetic order, and in the other half, they are in the reverse alphabetic order. All triples where the two candidates are in the reversed alphabetic order are now removed. The remaining triples are as follows:

$$\{Alice, Bravo, \theta_1\}\ \{Alice, Charlie, \theta_1\}\ \{Alice, Delta, \theta_1\}$$
$$\{Alice, Echo, \theta_{-1}\}\ \{Bravo, Charlie, \theta_1\}\ \{Bravo, Delta, \theta_1\}$$
$$\{Bravo, Echo, \theta_{-1}\}\ \{Charlie, Delta, \theta_1\}\ \{Charlie, Echo, \theta_{-1}\}$$
$$\{Delta, Echo, \theta_{-1}\}$$

After all the received votes are interpreted in the above format, a pairwise comparison of every two candidates is done to check which candidate is more preferred by the voters. For example, to compare *Alice* and *Bravo*, the triple in every vote which contains these two candidates is selected. The third values of these triples is aggregated into one ciphertext using the additive homomorphic property. If this ciphertext is decrypted, a positive plaintext indicates that Alice is more preferred than Bravo, and a negative plaintext indicates the opposite. If there exists a candidate who wins every pairwise comparison, that candidate is elected. In case the tally does not result a winner, some other additional methods must be used to determine the winner.

[5] Note that in Paillier cipher, the plaintext space is Z_n so we cannot directly encrypt the plaintext -1, but we can encrypt $n-1$ instead.

5 System Analysis

In this section, we present an informal analysis of our proposed scheme.

- *Privacy and receipt-freeness:* The random candidate ordering provides voter privacy. If the left hand column as depicted in Figure 2 is detached and destroyed, the right hand column does not reveal how the voter voted. In addition, to see if our scheme provides receipt-freeness we must consider the Italian attack. As discussed earlier, if a ranked election contains a large number of candidates, adversaries can coerce the voter to cast the vote with a unique ordering of candidates, and then they can check whether this pattern has appeared in the list of cast ballots. Our scheme never reveals the entire plaintext vote and is therefore robust against this attack. A variant of the Italian attack can be found in [29]. For example, the adversary can coerce the voter to put a very unpopular candidate as the first preference and a very popular candidate as the second preference. In any round, if the unpopular candidate is eliminated but there is no transfer from this eliminated candidate to that popular candidate, the adversaries will know that the voter did not follow the instructions. In our scheme, the transfer history is kept secret during the vote processing phase and therefore, our scheme is robust against this variant of the Italian attack as well.
- *End-to-end verifiability:* In our scheme, voters can use the "cut-and-choose" method to check whether the ballots are correctly generated. This ensures that the voter's intention will be correctly encoded. Each voter is also provided with a receipt, and the voter can check that the vote has been recorded by the system. These two actions provide individual verifiability. As the entire vote processing phase can be publicly verified, our scheme also achieves universal verifiability.
- *Robustness:* We have ensured that the various steps in the vote processing phase can either be done by any party without requiring the knowledge of the secret key or by a threshold set of parties. Therefore, so long as there exists a quorum of honest threshold parties, the correct election result will always be generated.
- *Complexity:* Our scheme has been designed so that it handles both homomorphic tallying and mixnet based tallying. Hence we can tailor the design of the vote processing phase based on different election methods, so that the election result can always be output in the most efficient manner. In FPTP and Approval Voting elections, all received votes are tallied using the homomorphic property. Hence the election result can be generated very efficiently. In Supplementary Vote, IRV and STV elections, the received votes also can be tallied using the homomorphic property if the election winner(s) can be identified in the first round. Otherwise, we need to use mixnets in the vote processing phase, and this phase may contain several rounds. In Condorcet voting, we interpret each of the received vote into a number of data triples, where each data triple pairwise compares two candidates. This process is done by mixnets and its complexity is quadratic in the number of candidates.

– *Usability:* Voters only need to be involved in the vote capture phase, and their experience is simple and consistent for various election methods. Also, the ballot form in our proposed scheme is very similar to those used in current paper based elections around the world.

5.1 Open Problems

A number of avenues to improve our scheme exist and we list a few. Note that some of the identified problems are common to existing voting schemes.

– *Authority knowledge attack:* All ballots are generated by a single party. Hence we need to trust this party for privacy and receipt-freeness. Generating the ballots in a distributed fashion is desirable, because it ensures no one but the voter ever learns the candidate ordering. However, achieving distributed ballot generation is not easy in our scheme. There are three major obstacles: Firstly, how to prove the ballot is well-formed in the distributed fashion. Secondly, how to print the ballot without the printer(s) learning the candidate ordering. And thirdly, how to ensure robustness so that the scheme can be run even in the presence of some dishonest election officials. Solving one or two of the above obstacles might be possible, but it is still an open problem whether all these three obstacles can be solved at the same time.
– *Chain voting attack:* If adversaries can successfully smuggle a blank ballot form out of the polling station, they can use this ballot to coerce voters. They mark an initial ballot with the candidate of their choice and give it to a voter entering the polling station. If the voter emerges with another blank ballot, she is rewarded. The adversaries can repeat this attack with the next voter using the new ballot. Ryan and Peacock have discussed this attack in [26], and some of their countermeasures also work for our scheme.
– *Randomisation attack:* Adversaries can coerce voters to bring out their receipts with some unique pattern, e.g. the cross always at the top or the ranking is in the ascend order. Although they do not know how these voters have cast their votes, they effectively force the voter to vote randomly. In FPTP elections, there exists a countermeasure to foil the randomisation attack. If the voter was coerced to bring out her receipt with the cross at the top, she can keep auditing ballots until she receives one with her favorite candidate at the top. But in ranked elections, such a countermeasure is not so effective and the voters may still be coerced to cast their votes randomly.
– *Usability:* Voters with certain specific disabilities may not be able to tear the paper ballot apart along the perforation. In future works we hope to investigate how to improve the accessibility of our schemes for these voters.
– *Scalability:* Our scheme is not suitable for elections with a large number of candidates. Since the candidate list is in random order, voters may have difficulty to find candidates in a long candidate list, especially in ranked elections. Moreover, the size of the homomorphic counter will become very large if there are many candidates, and some of the building blocks will become inefficient.

6 Conclusion

We have introduced a generic end-to-end verifiable voting scheme that handles many of the currently used election methods. We believe our work is an important step forward from the voter's point of view, keeping the voting experience simple and consistent no matter what election method is employed. Our work is based on the success of many existing concepts and building blocks, and we also contribute several improvements to these previous works:

- Lundin [17] and Popoveniuc et. al. [23] have earlier discussed how end-to-end verifiable voting schemes can be designed using the modular approach. For example, they discuss how the front-end and back-end of Prêt à Voter [9], [27] and Punchscan [24] can be designed in a mix-and-match fashion. However, they only focus on a single election method. In this paper, we extend the concept to multiple election methods and illustrate that multiple back-ends, each for a different election method, can be designed in the modular fashion, with a unified front-end.
- We introduce a very simple and straightforward solution to the Shuffle-Sum protocol [5] to handle partial ranking, and we also show that the Shuffle-Sum protocol can be designed much more efficiently if the election result can be tallied in the first round.
- We introduce a novel tallying method for Condorcet elections. Compared to the scheme in [10], we have simplified the voter interface without introducing extra complexity in the tallying phase.

In the future, we hope to introduce further enhancements to mitigate the open problems.

7 Acknowledgement

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/G025797 and the FNR (National Research Fund) Luxembourg under project SeRVTS–C09/IS/06. We would also like to thank the anonymous reviewers for their comments on the paper.

References

1. Ben Adida and Andrew Neff. Efficient receipt-free ballot casting resistant to covert channel. *In EVT'09*, 2009.
2. Ben Adida and Ronald L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. *In WPES'06*, pages 29–40, 2006.
3. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. *In PODC'01*, pages 274–283, 2001.
4. Josh Benaloh. Towards simple verifiable elections. *In WOTE'06*, pages 61–68, 2006.

5. Josh Benaloh, Tal Moran, Kim Ramchen, Lee Naish, and Vanessa Teague. Shuffle-Sum: coercion resistant verifiable tallying for STV voting. *IEEE Transactions on Information Forensics and Security*, December, 2009.
6. Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo Voting: secure and coercion-free voting using a trusted random number generator. *In VOTE-ID 2007*, pages 111–124, 2007. LNCS 4896.
7. David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: end-to-end verifiable for optical scan election systems using invisible ink confirmation codes. *In EVT'08*, 2008.
8. David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan T. Sherman, and Poorvi Vora. Scantegrity: end-to-end voter verifiable optical-scan voting. *Journal of IEEE Security & Privacy*, 6:3:40–46, 2008.
9. David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. *In ESORICS'05*, pages 118–139, 2005. LNCS 3679.
10. Michael Clarkson and Andrew Myers. Coercion-resistant remote voting using decryption mixes. *In FEE 2005*, 2005.
11. Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditional secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. *In TCC'06*, pages 285–304, 2006. LNCS 3876.
12. Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. *In EUROCRYPT'01*, pages 152–165, 2001. LNCS 2045.
13. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. *In FC'00*, 2000. LNCS 1962.
14. Pierre-Alain Fouque and Jacques Stern. Fully distributed threshold RSA under standard assumptions. *In ASIACRYPT'2001*, 2001. LNCS 2248.
15. James Heather. Implementing STV securely in Prêt à Voter. *In CSF'07*, pages 157–169, 2007.
16. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. *In USENIX Security Symposium*, pages 339–353, 2002.
17. David Lundin. Component based electronic voting systems. *Proceedings of IAVoSS Workshop on Trustworthy Elections (WOTE 2007)*, pages 11–16, 2007. Ottawa, Canada.
18. Andrew Neff. Practical high certainly intent verification for encrypted votes. *VoteHere document*, 2004.
19. Lan Nguyen, Rei Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: a formal model and a Paillier-based efficient construction with provable security. *In ACNS'04*, pages 61–75, 2004. LNCS 3089.
20. Pascal Paillier. Public-key cryptosystems based on discrete logarithms residues. *In EUROCRYPT'99*, pages 223–238, 1999. LNCS 1592.
21. Kun Peng and Feng Bao. A design of secure preferential e-voting. *In VOTE-ID 2009*, pages 141–156, 2009. LNCS 5767.
22. Kun Peng, Colin Boyd, and Ed Dawson. Simple and efficient shuffling with provable correctness and ZK privacy. *In CRYPTO'05*, pages 188–204, 2005. LNCS 3621.
23. Stefan Popoveniuc and Poorvi Vora. A framework for secure electronic voting. *Proceedings of IAVoSS Workshop On Trustworthy Elections (WOTE'08)*, 2008. Leuven, Belgium.
24. Punchscan. http://www.punchscan.org.

25. Ronald L. Rivest and Warren D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. *Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*, 2007. Boston, MA.
26. Peter Y. A. Ryan and Thea Peacock. Threat analysis of cryptographic election schemes. *Technical Report of University of Newcastle*, CS-TR:971, 2006.
27. Peter Y. A. Ryan and Steve Schneider. Prêt à Voter with re-encryption mixes. *In ESORICS'06*, pages 313–326, 2006. LNCS 4189.
28. Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for paillier encrypted values. *In EUROCRYPT'06*, pages 522–537, 2006. LNCS 4004.
29. Vanessa Teague, Kim Ramchen, and Lee Naish. Coercion-resistant tallying for STV voting. *In EVT'08*, 2008.
30. Pei-Yih Ting and Xiao-Wei Huang. Distributed paillier plaintext equivalence test. *International Journal of Network Security*, 6(3):258–264, 2008.
31. Roland Wen and Richard Buckland. Minimum disclosure counting for the alternative vote. *In VOTE-ID 2009*, pages 122–140, 2009. LNCS 5767.
32. Zhe Xia, Steve Schneider, James Heather, Peter Y. A. Ryan, David Lundin, Roger Peel, and Philip Howard. Prêt à Voter: All-In-One. *In WOTE 2007*, pages 47–56, 2007.

Appendix

We briefly summarise the vote casting procedure and tallying details of the various election methods considered in this paper for reference.

- **First-Past-The-Post (FPTP):** In FPTP elections, the voter simply puts a mark next to her preferred candidate and the candidate with the most votes wins. FPTP is used in various elections in Canada, India, the UK and some elections in the US.
- **Approval Voting:** In Approval Voting, the voter can indicate multiple preferences up to a set maximum. All votes carry the same weight and the candidates with the most votes wins. Approval Voting is currently used in some local council elections in the UK.
- **Supplementary Vote:** In Supplementary Vote elections, the voter marks her ballot as follows: she first marks 1 next to her most favourite candidate. If she also has a second preference, she marks 2 next to this candidate. Tallying takes place in at most two rounds. In the first round, only the first preference on every ballot is taken into account. If some candidate receives more than half of the votes, the election ends and this candidate wins. If no candidate wins in the first round, all candidates apart from those in the first two places are eliminated. Now, ballots with the first preference for one of the eliminated candidates are checked for their second preferences. If a ballot does not have a second preference or its second preference is also for one of the eliminated candidates, it is discarded. Otherwise, its second preference is treated as its first preference. Now, one of the two non eliminated candidate with the most votes wins. Supplementary Voting is used to elect mayors in the UK (e.g. the London Mayor Elections) and a variant is used in the Sri Lankan presidential elections.
- **Instance Runoff Voting (IRV):** IRV is also sometimes called alternative vote or preferential voting. In IRV elections, voters rank candidates based on their preference and depending on the election, ranking all candidates may be mandatory or optional. The winner is determined by a quota, which is normally half of the

received votes. In the first round of tallying, only the first preference of every ballot is taken into account. If some candidate receives more votes than the quota, the candidate wins and the election ends. Otherwise, the candidate with the least votes will be eliminated and all ballots with the highest preference for this voter will be redistributed among the remaining candidates, based on the next preference. In case the next preference is empty or all the remaining preferences are for eliminated candidates, the ballot is discarded. The process is repeated until a winner is found. Currently, IRV is used in some local government elections in the US, New Zealand and Malta. Moreover, there are proposals to replace the FPTP system used for parliamentary elections in the UK with the IRV.

- **Single Transferable Vote (STV):** Unlike the other methods described so far, in STV, more than one candidate is elected. To cast a vote, the voter ranks as few or as many candidates as she likes. To be elected, a candidate must receive more votes than a set quota. In the first phase of tallying, only the first preference on every ballot is taken into account. If no one receives more votes than the quota, the candidate with the least votes will be eliminated, and all ballots for this candidate will be redistributed among the remaining candidates, based on the next preference. However, if some candidates receive more votes than the quota, they will be elected and it will be checked if all seats are filled. If yes, the election ends. Otherwise, the tally continues so as to fill the remaining seats. Elected candidates do not need votes they receive over the quota and the surplus votes are transferred to the remaining candidates based on the next preference[6]. The above process is repeated until all seats are filled. STV is a popular election method and is currently used to elect the lower house of parliament in some territories in Australia, in local government elections in Scotland as well as some elections in Northern Ireland.
- **Condorcet voting:** In Condorcet elections, the voter provides a full ranking of the candidates. This allows every candidate to be compared to every other candidate. For each pairwise combination, it is checked which candidate is more preferred by the voters and the election winner is the candidate who wins every pairwise combination. Note that this process does not always result in a winner and special methods are required to determine the winner, but these techniques are out of the scope of this paper. The Condorcet method is not used in political elections, but it can be found in popular media, e.g. elections organised by MTV.

[6] Note that there are several different methods to transfer the surplus votes. For example, fraction transfer is used in Australia and Scotland, and random transfer is used in Northern Ireland.