Analysis, Improvement and Simplification of Prêt à Voter with Paillier Encryption

Zhe Xia Steve A. Schneider James Heather
Department of Computing, University of Surrey, U.K.
{*z.xia, s.schneider, j.heather*}*@surrey.ac.uk*

Jacques Traoré
France Telecom, Orange Lab
jacques.traore@orange-ftgroup.com

Abstract

In this paper, we analyse information leakage in Ryan's Prêt à Voter with Paillier encryption scheme (PAV-Paillier). Our analysis shows that although PAV-Paillier seems to achieve a high level of voter privacy at first glance, it might still leak voter's choice information in some circumstances. Some threats are trivial and have appeared in the literature, but others are more complicated because colluding adversaries may apply combined attacks. Several strategies have been suggested to mitigate these threats, but we have not resolved all the threats. We leave those unsolved threats as open questions. In order to describe our analysis in a logical manner, we will introduce an information leakage model to aid our analysis. We suggest that this model can be applied to analyse information leakage in other complex mixnet based e-voting schemes as well.

Furthermore, we introduce a simplification of PAV-Paillier. In our proposal, without degrading security properties such as voter privacy, verifiability and reliability, we no longer need to apply the homomorphic property to absorb the voter's choice index into the onion, thus we step back to employ the ElGamal encryption. This results in a simpler and more straightforward threshold cryptosystem. Some other attractive properties of our proposal scheme are: unlike traditional Prêt à Voter schemes, the candidate list in our scheme can be in alphabetical order. Our scheme not only handles approval elections, but also it handles ranked elections (e.g. Single Transferable Voting). Furthermore, our scheme mitigates the randomisation attack.

1 Introduction

Over the past two decades, a number of cryptographic e-voting schemes have been introduced. Compared to traditional election methods, these schemes are not only more secure but also more transparent. Generally speaking, based on the assumption that all parties are limited to polynomial-bounded computational resources, majority cryptographic e-voting schemes are aiming to achieve a number of properties which can be classified into the following five aspects:

Voter privacy: this aspect includes *privacy* and *receipt-freeness*. The privacy property keeps the voter's choice information private from others if the voter does not want to reveal it. Receipt-freeness, introduced by Benaloh and Tuinstra [9], has a much stricter requirement than privacy. It requires that voters are unable to generate receipts to prove to others how they have voted, or that they can use fake receipts to lie to adversaries, and adversaries cannot distinguish fake receipts from genuine ones. Therefore, receipt-freeness implies privacy. In this paper, voter privacy will have the same security requirement as receipt-freeness. The purpose is to ensure that the whole election system will not leak voters' choice information, even if voters are coerced to collude with adversaries. Therefore voters can cast their votes without coercion and intimidation. We say that the election system suffers from the information leakage problem if some threats against voter privacy can be introduced with non-negligible probability.

Verifiability: this aspect includes *integrity*, *correctness*, *individual verifiability* and *public verifiability*. Integrity means that all eligible voters will be allowed to vote, each eligible voter is allowed to vote once and only once, and all valid votes will be tallied. Correctness means that if all participants in the election behave honestly, the correct result will always be obtained. Individual verifiability ensures a correct mapping from voters' intent to the received votes, and public verifiability ensures a correct mapping from the received votes to the final result. The verifiability aspect is a combination of the above four properties, therefore it ensures a correct mapping from voters' intent to the final result, and furthermore, this procedure can be verified.

Reliability: includes *robustness* and *fairness*. Robust-

ness requires that the election systems are able to tolerate some mistakes caused by ordinary voters, as well as some faulty behaviour caused by a minority of election authorities. The ability to recover from cheating is important as well. Fairness ensures that no partial result will be revealed before the final result is announced.

Versatility: the election system is able to handle different election methods, such as First-Past-The-Post elections, Borda Court elections, Condorcet elections and Single Transferable Voting.

Usability: the election system can be used by ordinary voters without special knowledge (e.g. click-and-go). Besides, the system should be easy for election authorities to set-up and control, no special or expensive equipment is needed. Furthermore, the system can be executed efficiently even if the number of candidates or number of voters scales up.

Among these aspects, the first three, which are called *security aspects*, are always crucial, that particularly of certain for cryptographic electronic voting. The other two aspects are called *auxiliary aspects*. And there are two major conflicts among these aspects:

1. Voter privacy v.s. verifiability
2. Security aspects v.s. auxiliary aspects

1.1 Voter privacy v.s. verifiability

Generally speaking, early cryptographic e-voting schemes were focused on the ballot tallying phase, aiming to solve the first conflict by achieving voter privacy and verifiability simultaneously. These schemes can be classified into two major approaches: *based on mixnets*, and *based on homomorphic encryption*.

In schemes based on mixnets, each voter first encrypts her choice under some public key. The corresponding secret key is distributed among a number of decryption parties in a threshold fashion [46, 26]. The voter publishes this encrypted value on the bulletin board with some proof to show her knowledge of the plaintext. The proof aims to prevent the ballot duplication attack in [47]. When all voters have cast their votes, a number of mix servers will launch the shuffle phase. Each mix server receives a batch of encrypted votes, randomly re-encrypts and shuffles them, then outputs the result to the next mix server. Therefore, if all mix servers are honest, the outputs of the shuffle phase will be a permutation of the incoming votes, but their relationships have been mixed to ensure voter privacy. Finally, each of the encrypted values in the outcome will be decoded in a threshold fashion. Each decryption party publishes some auxiliary information [13] to prove the correctness of decoding. In order to audit the shuffle phase, each mix server needs to post some certificates[1] onto the bulletin board which can be verified by the public afterwards. These certificates should not reveal how the mix server has shuffled the encrypted values. Note that the most expensive part in the mix networks is for mix servers to prove the shuffle. There have been some attempts to improve efficiency. Some repetitive robust mixnets are interesting, e.g. [31, 32, 27], but most of them have been broken [19, 39, 1, 55]. We advocate choosing one from [25, 41, 28][2].

Schemes based on homomorphic encryption were first introduced by Benaloh [15, 10, 7]. To cast a vote, each voter generates an encrypted value of her desired vote and posts it onto the bulletin board. A proof that her encrypted value contains a valid vote is also required. The proof should not reveal the content of that vote, but it prevents dishonest voters from casting multiple votes. The received encrypted values are then aggregated in some public manner. Finally, the result can be tallied in a threshold fashion without opening each ballot for voter privacy. Similarly, some certificates to prove the correct decoding are published on the bulletin board. Schemes based on homomorphic encryption are efficient in obtaining the final result[3], however intensive zero-knowledge proofs are needed to prove the validity of each encrypted vote, and voting is normally restricted to the 1-out-of-L format. Major issues and important building blocks of e-voting schemes based on homomorphic encryption have been introduced in [17, 6, 18].

In these early schemes, the receipt-free property has attracted a lot of interest. Each encrypted vote is published, and the voter herself can prove the content of her vote by revealing the randomness she used for encryption. Thus some additional methods need to be applied in order to achieve receipt-freeness. We have to note that although a number of schemes have claimed achievement of the receipt-free property, some of them have been broken or shown to suffer drawbacks.

(1). Some schemes [9, 37, 38] suggest that after the voter encrypts her vote, some trusted party will re-encrypt each voter's vote and prove the re-encryption in an interactive way. However, if adversaries force voters to use some special randomness for their challenge,

[1] Note that the interactive proofs can be transformed into non-interactive ones using Fiat-Shamir heuristics [22], and some early schemes [11, 45] can be made publicly verifiable by applying [34].

[2] Note that [25] has been shown not to achieve computational zero-knowledge in [24], but it is proved that their mixnets will not reveal the relationships between inputs and outputs.

[3] Note that the decryption of exponential ElGamal ciphertexts in some schemes, e.g. [16, 17], still suffers drawbacks. But compared with schemes based on mixnets, schemes based on homomorphic encryption only need to decrypt one ciphertext instead of decrypting each received votes separately.

e.g. the outcome of some hash functions, then receipt-freeness fails. Therefore, the re-encryption proof has to be given in a non-interactive and designated-verifier way.

(2). There are two reasons why if an e-voting system uses designated verifier proof (DVP) [35] to prove some facts, the prover needs to be some trusted party, e.g. a tamper-resistant device, or some distributed parties [5]. First, when receiving some DVP proof, the verifier cannot make an accusation if she is not satisfied with the proof. Second, if adversaries coerce a voter to surrender her private key and then share the key with the prover, the prover can generate fake proofs which will be accepted by the voter. Furthermore, the verifier needs to prove knowledge of her private key [53]. Otherwise, the designated verifier proof will be meaningless to her.

(3). Hirt and Sako [30] claimed that a one-way untappable channel from the authorities to the voters to achieve receipt-freeness has the weakest physical assumption. However, this kind of structure (also in [52]) is not very elegant. Authorities prove the shuffle of encrypted votes using DVP, and receipt-freeness is achieved if voters are able to lie to adversaries about the shuffle. However, in the context of coercion, the possibility of a lying voter being caught is linear in the number of authorities colluding with the coercer.

1.2 Security aspects v.s. auxiliary aspects

The early cryptographic e-voting schemes introduced above require all voters to have an advanced knowledge of mathmatics. Obviously, this requirement is not realistic for ordinary voters. Some suggestions are for voters to indicate their intent to some voting device (e.g. DRE machine) or election authorities. These parties then help the voters to generate the encrypted votes and publish them onto the bulletin board. However, this might result in two problems:

1. How can voters ensure that their intent has been correctly transferred into encrypted votes, and these votes are recorded in the election system?

2. How can voters ensure that the voting device or election authorities, who know their intent, will not leak the information?

Two approaches, SureVote [12] and MarkPledge [42], that aim to solve the first problem have been proposed by Chaum and Neff respectively. Both systems provide voters with a receipt, which does not allow voters to prove how they have voted, but enables them to verify that their votes have been recorded in the election system. To ensure that the voter's intent has been correctly contained in the receipt, Chaum and Neff have introduced different ideas.

In Chaum's scheme [12], the voter indicates her intent to the voting machine, which then prints the ballot into two layers that are encoded using visual cryptography. The voter can read her intent on the ballot when the two layers are laminated together. After the voter has approved the ballot, she separates the two layers, randomly chooses one layer to retain as a receipt and destroys the other part. Later, Ryan's Prêt à Voter system [14] simplified Chaum's method using cut-and-choose. In [14], each voter will be provided with two ballots, she can randomly choose one ballot to challenge and use the other to cast her vote. If the ballot is correctly generated, the receipt will contain voter's intent. Although a single voter only has 50% chance to detect the fraud ballot, any attempt to cheat in more than a very small number of ballots would be detected with overwhelming probability. Ryan's cut-and-choose method also appears in [51, 4, 8].

Neff's scheme [42] is based on a different approach. The voter first indicates her intent to the voting machine, the machine then constructs an encrypted electronic ballot representing this voter's intent and commits it. After that, the voter interacts with the voting machine to obtain a receipt. Note that the voter's task in this process can be simplified, therefore ordinary voters can cast their votes without special knowledge, and some *helper organisations* [2] can help to verify the receipt afterwards. Similar idea of voter intent verification can be found in [3, 40].

The schemes introduced above have only solved our first problem. However, because voters need to indicate their intent to some voting device or election authorities, the voter privacy will be violated if the voting device or election authorities are faulty. Some work of e-voting threat analysis [36, 50] have shown that because of this problem, voter's choice information can be leaked in a number of ways. Taking the Prêt à Voter scheme [14] as an example, because all ballot forms are generated by some election authorities in advance, these authorities have the ability to read the voter's choice directly from their receipts. Besides, these authorities can apply *subliminal & Kleptographic channel attacks* to enable their colluding parties to read voter's receipt as well. Furthermore, the *Chain-of-custody* issues require that the ballots generated in advance cannot be tampered with before use, e.g. during transmission. Otherwise, voter privacy will be violated because of information leakage.

To the best of our knowledge, the first e-voting scheme which solves both our problems is Ryan's Prêt à Voter with re-encryption mixes [51], in which all ballots are generated by a number of election authorities, called *clerks*, in a distributed fashion. Therefore, although some voting device is still proposed, the voters no longer need to indicate their intent to the voting device or some sin-

gle clerk. In theory, voter privacy can be computationally preserved under the assumption that all clerks do not collude. However, in order to apply the homomorphic property to absorb voter's choice index into the onion, [51] has employed the exponential ElGamal encryption, which is not a trapdoor function. After decryption, we need to search some large field in order to retrieve the plaintext. To overcome this problem, Ryan has introduced Prêt à Voter with Paillier encryption (PAV-Paillier) [49, 48].

1.3 Our contribution

Generally speaking, the PAV-Paillier scheme achieves a high level of security properties. It is reliable because the shuffle phase and the decryption phase are separated, and the decryption phase is executed in the threshold fashion. It is end-to-end verifiable because each voter can verify that her vote has been properly recorded, and anybody can verify that all recorded votes are correctly tallied. Compared with other schemes, PAV-Paillier is remarkably superior for ensuring voter privacy. The relationships between encrypted votes and results are shuffled in the ballot tallying phase, and in the ballot casting phase, instead of trusting a single voting device or some election authorities, we just need to trust that there exists at least one honest clerk.

In this paper, we analyse information leakage in PAV-Paillier. Our analysis shows that although PAV-Paillier seems to achieve a high level of voter privacy at first glance, it might still leak voter's choice information in some circumstances. Some threats are trivial and have appeared in the literature, but others are more complicated because colluding adversaries may apply combined attacks. Several strategies have been suggested to mitigate these threats, but we have not resolved all the threats. We leave those unsolved threats as open questions. In order to describe our analysis in a logical manner, we will introduce an information leakage model to aid our analysis. We suggest that this model can be applied to analyse information leakage in other complex mixnet based e-voting schemes as well.

Furthermore, we introduce a simplification of PAV-Paillier. In our proposal, without degrading security properties such as voter privacy, verifiability and reliability, we no longer need to apply the homomorphic property, thus we step back to employ the ElGamal encryption. This results in a simpler and more straightforward threshold cryptosystem. Some other attractive properties of our proposal scheme are: unlike traditional Prêt à Voter schemes, the candidate list in our scheme can be in alphabetical order. Our scheme handles both approval elections and ranked elections (e.g. Single Transferable Voting). Furthermore, our scheme mitigates the randomisation attack.

1.4 Outline of this paper

The paper is organised as follows. In Section 2, we introduce a model which is helpful to analyse information leakage in complex e-voting schemes. The PAV-Paillier scheme will be briefly reviewed in Section 3, and it will be analysed in Section 4. A simplification of PAV-Paillier will be introduced in Section 5. Finally, we conclude in Section 6.

2 Information Leakage Model

In this section, we first introduce some terminologies for the information leakage model. Then we illustrate how this model can be applied to analyse information leakage in a simple e-voting scheme.

2.1 Terminologies

Definition 1 (Relation) *Let $\mathcal{X}$ and $\mathcal{Y}$ are finite sets, and f be a bijection from $\mathcal{X}$ to $\mathcal{Y}$ such that f is kept private. We say that there is a relation from $\mathcal{X}$ to $\mathcal{Y}$, if for any particular target element $x \in \mathcal{X}$, with non-negligible probability, its image $y = f(x)$ in $\mathcal{Y}$ can be found. We denote such a relation as $\mathcal{X} \Longrightarrow \mathcal{Y}$.*

Definition 2 *If there exists a relation from $\mathcal{X}$ to $\mathcal{Y}$ as well as a relation from $\mathcal{Y}$ to $\mathcal{X}$, we say that there exists a relation between $\mathcal{X}$ and $\mathcal{Y}$. We denote such a relation as $\mathcal{X} \Longleftrightarrow \mathcal{Y}$.*

Corollary 1 (Relation is transitive) *Suppose $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are finite sets, f is a bijection from $\mathcal{X}$ to $\mathcal{Y}$, and g is a bijection from $\mathcal{Y}$ to $\mathcal{Z}$. If there exist a relation from $\mathcal{X}$ to $\mathcal{Y}$ and a relation from $\mathcal{Y}$ to $\mathcal{Z}$, then there is a relation from $\mathcal{X}$ to $\mathcal{Z}$.*

Proof. Since there is a relation from $\mathcal{X}$ to $\mathcal{Y}$, for a target element $x \in \mathcal{X}$, with non-negligible probability, its image $y = f(x)$ in $\mathcal{Y}$ can be found. Similarly, for the target element $y \in \mathcal{Y}$, with non-negligible probability, its image $z = g(y)$ in $\mathcal{Z}$ can be found. Then for a target element $x \in \mathcal{X}$, with non-negligible probability, its image $z = g(f(x))$ in $\mathcal{Z}$ can be found. Therefore, there is a relation from $\mathcal{X}$ to $\mathcal{Z}$.

Definition 3 (Information leakage) *In an election scheme based on mixnets*[4]*, $VOTER$ is the set which contains a list of eligible voters, and $RESULT$ is the*

[4]The model is not suitable for schemes based on homomorphic encryption because in those schemes, all received votes will be aggregated before the decryption, thus there is just a final result instead of a set of results for each vote.

set which contains a list of the final votes for tallying. In the ideal case[5]*, there will be some function f which is a bijection from $VOTER$ to $RESULT$, but for voter privacy, f is kept private. We say that the election scheme suffers the information leakage problem if there exists a relation from $VOTER$ to $RESULT$.*

Definition 4 (Threat) *A threat is a way information leakage can be introduced into an election scheme.*

Definition 5 (Attack) *An attack is a possible way of finding a relation from one set to another set.*

2.2 An E-Voting Example:

In a verifiable e-voting scheme based on mixnets, to cast a vote, a voter indicates her intent to the voting machine, and the machine generates an encrypted vote for this voter. Note that the cut-and-choose method can be used to challenge the voting machine. The voter submits her vote to the election authorities. These authorities record the encrypted vote and sign it. The voter can keep the signed vote as a receipt, which can be used to check that her vote has been recorded. Later, all received votes are shuffled by mixnets, and finally the results are decrypted and published onto the bulletin board.

In the above scheme, there will be three sets with the same cardinality: the set of voters, the set of receipts, and the set of result. We denote them as $VOTER$, $RECEIPT$ and $RESULT$ respectively. Based on our definition, the scheme will suffer information leakage if there is a relation from $VOTER$ to $RESULT$. Therefore, there might be two possible threats of information leakage in the scheme, as shown in Figure 1.

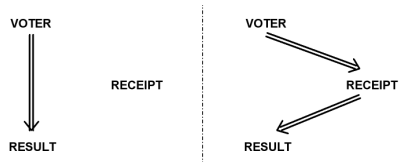

Figure 1: Two threats of information leakage

Threat 1: $VOTER \Longrightarrow RESULT$

Some attacks may be applied by adversaries to generate a relation from $VOTER$ to $RESULT$:

(1). *Hidden camera attack:* if the voting booth is monitored by hidden cameras, adversaries can directly know the choice made by a target voter.

(2). *Knowledge of the voting machine:* since the voting machine learns the voter's intent, a cheating voting machine can leak the choice made by a target voter.

[5]The ideal case means that each eligible voter will cast one and only one valid vote, and each valid vote will be counted.

(3). *Italian attack:* this attack only happens in ranked elections with a large number of candidates. Adversaries can coerce a target voter to cast her vote in some particular manner. The voter will be caught with high possibility if she does not cooperate. This attack is possible when the number of possible votes are much more than the number of voters.

Threat 2: $VOTER \Longrightarrow RECEIPT \Longrightarrow RESULT$

Each voter will be provided with a receipt in the above scheme, and any voter may reveal her receipt to others. Therefore, there is always a relation from $VOTER$ to $RECEIPT$. The remaining task for an adversary to violate voter privacy is to generate a relation from $RECEIPT$ to $RESULT$. Several attacks can be applied to generate such a relation:

(1). *Ballot duplication attack:* the attack is possible if the vote is not encrypted using non-malleable encryption [20, 31]: adversaries can cast a duplication of a target vote in order to find out the choice information of the target vote. For more information, see [47, 23].

(2). *Subliminal & Kleptographic channel attack:* the voting machine which generates the encrypted votes can employ this attack to enable any party colluding with it to read a voter's receipt without the need to have it decrypted.

(3). *Chain voting attack:* the above election scheme does not suffer this attack, but some schemes which uses paper ballots generated in advance may suffer this attack, e.g. the Prêt à Voter scheme [14]. If adversaries successfully smuggle one blank ballot out of the voting booth, they can force a target voter to cast her vote using this ballot and bring back another blank ballot. Therefore, the receipt tells the adversaries how this voter has voted.

The above election scheme is simple. Its use here is to illustrate how the model can be applied to analyse information leakage in mixnets based e-voting schemes. Note that all the listed attacks have been introduced in the literature, e.g. [36, 50]. But it is obvious that the information leakage model can help to summarise these attacks in some logical manner, and this is very important if the election scheme becomes complicated.

3 Review of PAV-Paillier

We now present an overview of the PAV-Paillier scheme [49, 48]. In the voting booth, each voter selects a ballot form at random. An example of the ballot form is shown in Figure 2.

The ballot form contains two columns with a perforation down the middle. There are two encrypted values, called *onions*, printed at the bottom. The onion in the

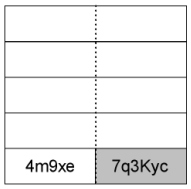

Figure 2: A blank ballot form

left hand column is called the booth onion, which can be decoded by the voting machine. If it is decrypted, the voting machine will derive the seed value and print the candidate list on the ballot form. At this moment, the ballot form will be shown in Figure 3. Note that the order of the candidate list is cyclically shifted and varies among different ballots, thus it is infeasible to predict the candidate order of a certain ballot form. The onion in the right hand column, which is printed on some scratch strips, is called the ready onion.

Figure 3: A ballot form with candidate list

3.1 Ballot construction

Paillier cipher [44] 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. The public key is (g, n), and the corresponding secret key $\lambda(n) = lcm((p-1),(q-1))$. To encrypt a message $s \in Z_n$, we randomly choose $x \in Z_n^*$ and compute the ciphertext $c = g^s x^n \pmod{n^2}$. To decrypt c, we compute $s = L(c^{\lambda(n)} \bmod n^2)/L(g^{\lambda(n)} \bmod n^2) \bmod n$, where the L-function takes in input values from the set $S_n = \{u < n^2 | u = 1 \bmod n\}$ and computes $L(u) = \frac{u-1}{n}$.

Construction of the receipt onions Suppose the public key (g, n) is made public and the corresponding secret key $\lambda(n)$ is shared among a number of decryption parties, called *tellers*, in a threshold fashion [54], where a quorum Q of these tellers work together can decrypt a ciphertext which is encrypted under the public key (g, n), but fewer than Q tellers will learn nothing from the ciphertext.

The first step to construct the ballot forms is for a set of $l+1$ clerks to generate a receipt onion for each ballot form.

The first clerk C_0 randomly chooses some seed values $s_i^0 \in Z_n$ and some blinding factors $x_i^0 \in Z_n^*$. Then C_0 generates a batch of initial onions as:

$$E(s_i^0, x_i^0) = g^{s_i^0}(x_i^0)^n \pmod{n^2}$$

After that, the remaining l clerks perform as follows: the jth clerk C_j receives a batch of onions $E(s_i^{j-1}, x_i^{j-1})$ from the clerk C_{j-1}. C_j chooses some seed values $\bar{s}_i^j \in Z_n$ and some blinding factors $\bar{x}_i^j \in Z_n^*$, and then she calculates

$$E(s_i^j, x_i^j) = E(s_i^{j-1}, x_i^{j-1}) \cdot E(\bar{s}_i^j, \bar{x}_i^j) \pmod{n^2}$$

where

$$s_i^j = s_i^{j-1} + \bar{s}_i^j \pmod{n}$$
$$x_i^j = x_i^{j-1} \cdot \bar{x}_i^j \pmod{n^2}$$

C_j shuffles the batch of onions $E(s_i^j, x_i^j)$ and then outputs them to the next clerk C_{j+1}.

Finally, the last clerk C_l will output a batch of onions $E(s_i, x_i) = g^{s_i}(x_i)^n \pmod{n^2}$ where

$$s_i = s_i^l = s_i^0 + \sum_{j=1}^{l} \bar{s}_i^j \pmod{n}$$

$$x_i = x_i^l = x_i^0 \cdot \prod_{j=1}^{l} \bar{x}_i^j \pmod{n^2}$$

We call the onions output by the last clerk *receipt onions*, in which each clerk has contributed to the entropy of the seed values from which the candidate list is derived. This process does not reveal the seed values. Therefore, the final seed values s_i can be kept private unless all these clerks collude. For each ballot, a unique receipt onion will be printed on it, at the bottom in the right hand column.

Transforming receipt onions into booth onions When some of the above ballot forms are generated, a number of re-encryption parties (which can be the same clerks) will perform the following tasks:

1. The party P_j collects the ballot forms from P_{j-1}, and P_j re-encrypts the onions in each ballot form as $E(s_i, x_i') = E(s_i, x_i) \cdot {t_i}^n$, where $x_i' = x_i \cdot t_i$, and t_i is randomly chosen from Z_n^*.

2. For each ballot form, P_j covers the original onion $E(s_i, x_i)$ with a scratch strip and overwrites it with its re-encrypted value $E(s_i, x_i')$.

3. P_j shuffles all ballot forms and then sends them to the next party P_{j+1}.

When the above procedure is finished, we call the onions $E(s_i, r_i)$ which are printed on top of the scratch surface *ready onions*. The major purpose of the above procedure is to break the links between the receipt onions and the ready onions. At this moment, we have a batch of ballot forms, each contains a unique receipt onion, but it is covered by multi-layer of scratch strip. The only visible value is the ready onion, which is the re-encryption of the receipt onion. Note that Ryan has suggested that in some cases, just one such re-encryption might be sufficient.

The ready onions are transformed into the booth onions, which can be directly decrypted by the voting machine. However, the transformation procedure introduced in PAV-Paillier is not very elegant. Ryan suggests that each ready onion is partially decrypted by a $Q-1$ subset of the tellers, and the result is the *booth onion*. In the ballot forms, the booth onion is printed at the bottom in the left hand column. Therefore, if the voting machine is provided with another share of the secret key $\lambda(n)$, it can decrypt the booth onion to obtain the seed value.

Auditing the ballot construction To audit the ballot construction phase, Ryan has introduced two methods in PAV-Paillier. The traditional cut-and-choose method, which is similar as in [14, 51], requires some tellers to be online. The other method aims to resolve this drawback, but some auxiliary information needs to be published. However, this information might result the leakage of the secret key $\lambda(n)$. Thus, the second method does not work. Because of space limitations, we will not describe this technical weakness in detail.

3.2 Ballot casting

When a voter is authenticated in the voting booth, she will be provided with a ballot form as shown in Figure 2. She then inserts the left hand column into the voting machine, which will read the booth onion, decode it, and print the corresponding candidate list on the ballot form. In this process, the seed value s_i will be retrieved first, then the candidate ordering is determined by $s_i \pmod{v}$, where v is the number of candidates. At this point, the ballot will be as in Figure 3.

This voter marks her choice, followed by tearing the ballot form apart along the perforation and destroying the left hand column. The voter brings the right hand column to the election officials and removes the scratch surface in their presence. After that, the voter can keep the right hand column as the receipt once it has been scanned by the election officials. A receipt example is shown as in Figure 4. The receipt can be used to check that the voter's vote has been properly recorded by the election system. We note that the receipt only contains the voter's mark and the receipt onion.

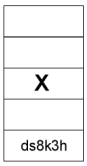

Figure 4: A receipt example

3.3 Ballot tabulation

After the election day, all received votes are collected. For each vote, the election officials will first perform some calculation to absorb voter's choice index value ι into the receipt onion as[6]

$$c' = (g^{s_i}(r_i)^n) \cdot g^{\iota} = g^{s_i+\iota}(r_i)^n \pmod{n^2}$$

Then these encrypted values (pure Paillier terms) will be inserted into some re-encryption mixnet, which will shuffle and re-encrypt these terms by changing the randomisations while leaving the seed values untouched. The mixnet can be audited using techniques in [34, 43]. Finally, the outputs of the mixnet will be decoded by a quorum of tellers in a threshold fashion, and the results will be announced.

4 Threat Analysis of Information Leakage in PAV-Pailler

At first glance, the PAV-Paillier scheme should provide a high-level of assurance of voter privacy based on the following three assumptions:

- In the entire scheme, more than a quorum of tellers are honest.
- In the ballot casting phase, there exists at least one honest clerk.
- In the ballot tallying phase, there exists at least one honest mix server.

In this section, we analyse information leakage in PAV-Pailler. Our analysis shows that even under the above three assumptions, a voter's choice information still might be leaked in a number of ways.

[6]Note that our computation of absorbing the choice index into the receipt onion is slightly different from Ryan's method. Ryan defines the cyclic shift direction as downwards, while we define it as upwards. Our change does not affect the threat analysis in the next section.

4.1 Threat analysis

There are four objects in the PAV-Paillier scheme: the voter, the ballot form (as shown in Figure 2), the receipt (as shown in Figure 4), and the result. In the ideal case, the sets of these four objects will have the same cardinality. We denote these sets as $VOTER$, $BALLOT$, $RECEIPT$, and $RESULT$ respectively. Based on our definition, the PAV-Paillier scheme will suffer the information leakage problem if there exists a relation from $VOTER$ to $RESULT$. Therefore, if we apply the information leakage model with PAV-Paillier, as shown in Figure 5, the following five possible threats might violate voter privacy. We will consider each in turn.

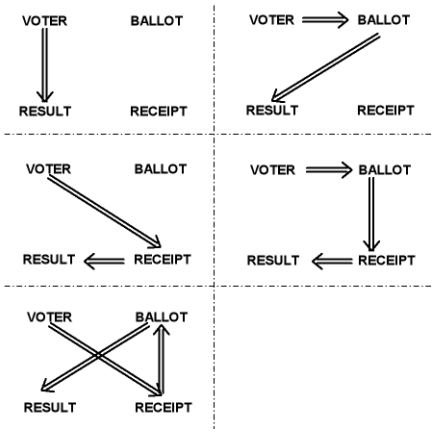

Figure 5: Five threats of information leakage

Threat 1: $VOTER \Longrightarrow RESULT$

We will examine some existing attacks to see whether the PAV-Paillier scheme suffers from this threat.

(1). *Italian attack:* PAV-Paillier does not suffer this attack because it cannot directly handle ranked elections.

(2). *Knowledge of the voting machine:* In PAV-Pailler, the voter does not indicate her intent to the voting machine. Therefore, the voting machine itself has no way to find out how this voter has voted.

(3). *Hidden camera attack:* PAV-Paillier suffers this attack, but all other election scheme will suffer this attack as well. To ensure voter privacy, the assumption is needed that the voting booth is not monitored by hidden cameras.

In the rest of this paper, we will not consider the hidden camera attack, and we suppose there is no direct relation from $VOTER$ to $RESULT$ in the PAV-Paillier scheme.

Threat 2: $VOTER \Longrightarrow BALLOT \Longrightarrow RESULT$

Within this threat, we will first check whether there is a relation from $BALLOT$ to $RESULT$. Based on our definition, such a relation can be generated if for any particular target ballot form, with non-negligible probability, the adversaries can find out how this ballot has been used to cast a vote.

In the PAV-Paillier scheme, for a target ballot form, the voting machine can learn the seed value s_i after the booth onion has been decrypted. In the ballot tallying phase, the receipt onion $c = g^{s_i}(r_i)^n \pmod{n^2}$ will absorb the voter's choice index ι as

$$c' = (g^{s_i}(r_i)^n) \cdot g^{\iota} = g^{s_i+\iota}(r_i)^n \pmod{n^2}$$

Then c' will be shuffled by the mixnet. The outcome of this ciphertext will be decrypted and the result $s_i + \iota$ will be published on the bulletin board[7]. Because $\iota < v$, where v is the number of candidates, with non-negligible probability, the voting machine will find that there is a unique value in the final result, whose difference to s_i is smaller than v. Therefore, the voting machine can learn how a ballot form has been used to cast a vote, and there exists a relation from $BALLOT$ to $RESULT$.

To violate voter privacy, the remaining task for adversaries is to find a relation from $VOTER$ to $BALLOT$. It is obvious that if some adversaries know which ballot form has been assigned to a particular voter, these authorities colluding with the voting machine can find out how this voter has voted. To overcome this problem, all ballot forms are required to be distributed in an anonymous way.

Furthermore, even if we suppose that all ballot forms are distributed in an anonymous way, and there does not exist a direct relation from $VOTER$ to $BALLOT$, does this mean that the PAV-Paillier can resist this threat? Unfortunately, the adversaries can generate the relation from $VOTER$ to $BALLOT$ as shown in Figure 6.

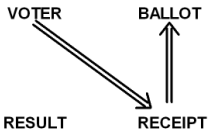

Figure 6: Generate a relation from $VOTER$ to $BALLOT$

Because any voter may reveal her receipt to others, there always exists a relation from $VOTER$ to $RECEIPT$. Therefore, the task to generate a relation from $VOTER$ to $BALLOT$ can be transferred to the task of generating a relation from $RECEIPT$ to

[7]Note that even if $s_i + \iota$ is not published on the bulletin board, all tellers who have been involved to decrypt it have known this value, and any dishonest teller can reveal this value.

$BALLOT$. In order words, if adversaries can generate a relation from $RECEIPT$ to $BALLOT$, their collusion with the voting machine can violate voter privacy. We will discuss whether there exists such a relation later.

Threat 3: $VOTER \Longrightarrow RECEIPT \Longrightarrow RESULT$

Because any voter may reveal her receipt to others, the relation from $VOTER$ to $RECEIPT$ always exists. The remaining task for adversaries to violate voter privacy is to generate a relation from $RECEIPT$ to $RESULT$. We will examine some existing attacks against this relation and check whether they can be applied with the PAV-Paillier scheme:

(1). *Chain voting attack:* PAV-Paillier does not suffer the chain voting attack because even if some adversaries can smuggle a ballot form (with the candidate list) out of the voting booth, they cannot coerce a voter to cast her vote using this ballot form. In one aspect, if the adversaries remove the scratch surface, the election officials will reject this vote when a coerced voter submits it. In the other aspect, if the adversaries keep the scratch surface intact, they will not know whether the voter has voted using this ballot form, because the relationship between a ballot form and its receipt is kept private thanks to the scratch surface.

(2). *Subliminal & Kleptographic channel attack:* The PAV-Paillier scheme is designed to resist this attack by generating the ballot forms in a distributed fashion. However, some mechanisms are necessary to ensure this. Otherwise, once a receipt onion is generated, no one can tell whether it is generated by a set of clerks or just by a single clerk. Therefore, if some receipt onions are generated by a single clerk, this clerk has the ability to learn some voters' choice just by reading their receipts. Also, this clerk can apply the subliminal & kleptographic channel attack to enable her colluding parties to learn these voters' choice just from their receipts.

(3). *Ballot duplication attack:* Generally speaking, the ballot duplication attack can be frustrated if the party who generates the encrypted value has proven the knowledge of the plaintext. However, in the PAV-Paillier scheme, the receipt onions are generated in a distributed fashion where each clerk contributes to the entropy of the seed values from which the candidate list is derived. The seed values will be kept private unless all these clerk collude. Therefore, the non-malleable encryption solution is not suitable for PAV-Paillier since it is infeasible for these clerks to generate such a proof. Another solution to this attack is that every receipt onion is proved to be generated by a number of authorised clerks. Thus if there exists at least one honest clerk, although some receipt onions might collide to have the same seed value, no receipt onion will be the duplication of others. Similar to the previous attack, some mechanisms are necessary to ensure this point.

Therefore, to avoid a direct relation from $RECEIPT$ to $RESULT$, all the receipt onions need to be proved that they have been generated by a number of clerks. One possible solution is that each clerk who has been involved in generating the receipt onion has to sign it. Thus, both the subliminal & Kleptographic channel attack and the ballot duplication attack can be frustrated. However, this still does not guarantee that the relation from $RECEIPT$ to $RESULT$ cannot be generated. As shown in Figure 7, if adversaries can find a relation from $RECEIPT$ to $BALLOT$ and a relation from $BALLOT$ to $RESULT$, they can still generate a relation from $RECEIPT$ to $RESULT$.

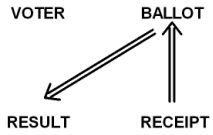

Figure 7: Generate a relation from $RECEIPT$ to $RESULT$

We have shown in *Threat 2* that the voting machine can generate a relation from $BALLOT$ to $RESULT$. Therefore, if any adversaries can generate a relation from $RECEIPT$ to $BALLOT$, the collusion of these adversaries and the voting machine can find out how a particular voter has voted. We will discuss whether the adversaries can generate a relation from $RECEIPT$ to $BALLOT$ later.

Threat 4: $VOTER \Longrightarrow BALLOT \Longrightarrow RECEIPT \Longrightarrow RESULT$

This threat contains three parts. The first part is to find a relation from $VOTER$ to $BALLOT$. We have shown in *Threat 2* that even if all ballot forms are distributed in an anonymous way, adversaries still can generate such a relation as shown in Figure 6. The second part is to find a relation from $BALLOT$ to $RECEIPT$. The third part of this threat is to find a relation from $RECEIPT$ to $RESULT$. Similarly, even if we apply some mechanisms to ensure that all ballot forms are generated in a distributed fashion, adversaries can still find such a relation as shown in Figure 7. When we combine these three parts together, we know that if adversaries can find a relation between $RECEIPT$ and $BALLOT$, the colluding of these adversaries and the voting machine can cause information leakage. Note that the requirement within this threat is stricter than in *Threats 2 & 3*, instead of requiring one way relation, a two way relation is needed here. In other words, if the voter privacy in PAV-Paillier can be violated by *Threat*

4, it can also be violated by *Threats 2 & 3*.

Threat 5: $VOTER \Longrightarrow RECEIPT \Longrightarrow BALLOT \Longrightarrow RESULT$

In *Threat 2*, we have shown that the relation from $BALLOT$ to $RESULT$ can be independently generated by the voting machine. In this threat, because there always exists a relation from $VOTER$ to $RECEIPT$, if an adversary can generate a relation from $RECEIPT$ to $BALLOT$, the collusion of this adversary and the voting machine can generate the relation from $BALLOT$ to $RESULT$ in a more straightforward manner. For example, a voter has revealed her receipt. Then an adversary who has the ability to generate a relation from $RECEIPT$ to $BALLOT$ can find out the ballot form which has been assigned to this voter. As follows, the voting machine decrypts this ballot form and obtains the candidate ordering. Therefore, the candidate ordering and the choice mark in the receipt will reveal how this ballot form has been used to cast a vote. Similarly, the key point of this threat is whether there exists a relation from $RECEIPT$ to $BALLOT$.

4.2 How to link ballot form with its receipt

We will introduce several attacks which might be applied by the adversaries to generate a relation between $RECEIPT$ and $BALLOT$:

(1). *Single re-encryption:* The PAV-Paillier scheme has suggested that just one re-encryption of the receipt onion might be sufficient in some case. However, this will enable the re-encryption party to learn the relationship between the ballot form and its receipt. Furthermore, if this party colludes with the coercers, the PAV-Paillier scheme will suffer the *chain voting attack*.

(2). *Ready onion leakage:* In the PAV-Paillier scheme, in order to counter the chain voting attack and the chain-of-custody issues, to submit a vote, the voter is required to remove the scratch strip in the presence of the election officials. However, if some officials see the ready onion before it has been removed, they learn the relationship between the ballot form and its receipt.

(3). *Fingerprinting attack:* Adversaries can apply this attack to link a ballot form with its receipt. For every ballot form, the adversaries leave a unique fingerprinting on both columns. When the voting machine prints the candidate list on the left hand column, it will build a relation between the unique fingerprinting and the ballot form. And when a voter submits her vote, some cheating officials will build a relation between the unique fingerprinting and the receipt. Thus, if they work together, they can link a ballot form with its receipt. Similar problem will occur if voters leave their fingerprinting on the ballot forms.

(4). *Particular power of the last re-encryption party:* The last re-encryption party in PAV-Paillier will have some particular power. When she receives a batch of ballot forms from the previous party, she can remove all the scratch surface and read every receipt onion. After that, she generates multiple scratch layers for each ballot form and prints the ready onion (which might be just one re-encryption of the receipt onion) at the top of the scratch surface. In this way, she has generated the link between every ballot form and its receipt. And in most cases, this attack will go without being detected.

4.3 PAV-Paillier improvement

Based on the above threat analysis, the PAV-Paillier scheme has been shown to suffer several information leakage problems. In order to achieve better voter privacy, we will introduce several strategies to mitigate these problems.

(1). All ballot forms need to be distributed in an anonymous way. Otherwise, if the party who distributes the ballot forms colludes with the voting machine, they might find out how a voter has voted. We suggest that after all ballot forms have been generated, each ballot will be put into an envelope with uniform layout, and all these envelopes are shuffled before use.

(2). All the receipt onions need to be proven that they have been generated by a number of clerks. Otherwise, if they are just generated by a single clerk, this clerk can violate voter privacy either by subliminal & kleptographic channel attack or by ballot duplication attack. It is required that each clerk who has been involved in generating the receipt onion has to sign it.

(3). The re-encryption of the receipt onion needs to be executed by a number of different parties. Just one such re-encryption is not sufficient. And we need to trust that there is at least one honest re-encryption party.

(4). When the voter submits her vote, the election officials should not read the ready onion before it is removed. We suggest that once the ready onion has been transformed into the booth onion, it will be covered by another layer of scratch surface. Therefore, the booth onion can be kept private in the ballot casting phase.

(5). Normally, we do not need to worry about the fingerprinting attack. But if it is an issue we have to consider, we need to apply some mechanisms to ensure the ballot forms are fingerprinting free. For example, some help organisations in the voting booth can help to check that the ballot forms are free of fingerprinting (or digital watermarking) when it is assigned to the voters. And each voter can be provided with some special gloves to avoid leaving the fingerprinting on the ballot form.

(6). The use of multiple layers of scratch strips has a drawback that the last re-encryption party has more

power than the others. We are not clear how to resolve this drawback. It might be necessary to replace the use of multiple scratch strips by some other techniques.

(7). In the PAV-Paillier scheme, the voting machine will have very high likelihood of finding out how a ballot form has been used to cast a vote. This is because $n = pq$, where p and q are large primes, and the seed value s_i is randomly chosen from Z_n. We suggest that all clerks randomly choose the seed values from a smaller set Z_v, where v is the number of candidates[8]. This will give the voting machine less chance to find a unique value in the final result whose difference to s_i is smaller than v.

5 A Simplification of PAV-Paillier

5.1 Other issues of PAV-Paillier

Apart from the information leakage problems we have shown in the previous section, the following issues might be considered as drawbacks in PAV-Paillier.

(1). The process of transforming the ready onion into the booth onion is not very elegant. The booth onion is the partial decryption of the ready onion by $Q-1$ tellers. Although this enables the voting machine to decrypt the booth onion, anyone else who has another share of the secret key will have the same ability as the voting machine. Therefore, some of the information leakage attacks might be applied by other tellers as well.

(2). The candidate list in PAV-Paillier is a cyclic shift from the canonical order. This is an intelligent idea to generate secret ballot forms. But in some elections with large number of candidates, e.g. hundreds of candidates, voters might face difficulty in finding their preferred candidate on the ballot form. The situation will become much worse if the candidate order is randomly permuted instead of just cyclic shifted, e.g. [29, 56]. Because of this problem, election rules in some districts require that the candidate order has to be in alphabetical order.

(3). The PAV-Paillier scheme only handles approval elections. However, ranked elections, e.g. Single Transferable Voting, are widely used in a number of areas. It is clear that any e-voting system aiming at a large potential market needs to be able to deal with ranked elections.

(4). The PAV-Paillier scheme, as well as some other schemes, suffers from the *randomisation attack*. Adversaries can coerce the voter to take out a receipt with the mark at the top. Although, adversaries are unable to learn the content of this vote, they coerce the voter to vote in a random manner.

In this section, we will introduce a variant of the PAV-Paillier scheme, aiming to resolve the above four issues. Furthermore, because of the replacement of the Paillier cipher by the ElGamal cipher, the threshold cryptosystem in our proposal scheme will be simpler and more straightforward. The security properties of our proposal scheme are similar to those in PAV-Paillier.

[8]Note that to ensure every clerk has chosen the seed values from Z_v, a cut-and-choose method can be used to challenge them.

5.2 Building blocks

ElGamal cipher [21] Let p and q be two large primes such that $q|p-1$. We denote G_q as the subgroup of Z_p^* with order q. Let g be a generator of G_q. The secret key is an element $x \in Z_q$ and the corresponding public key is $y = g^x \pmod p$. In the following, we assume all arithmetic to be modulo p where applicable, unless otherwise stated. To encrypt a plaintext $m \in G_q$, we choose a random blinding factor $r \in Z_q$ and compute the ciphertext $E(m,r) = (G,M) = (my^r, g^r)$. Note that an ElGamal ciphertext is a pair of values of G_q. To decrypt an ElGamal ciphertext (G,M), we compute $m = G/M^x$. ElGamal is a probabilistic public-key encryption scheme, which is semantically secure if the decision Diffie-Hellman assumption holds in the group G_q.

ElGamal re-encryption Given an ElGamal ciphertext $(G,M) = (my^r, g^r)$, a party can efficiently compute a new ciphertext (G',M') that decrypts to the same plaintext as (G,M). We denote that the ciphertext (G',M') is a re-encryption of (G,M). To re-encrypt a ciphertext, the mix server chooses a value $s \in Z_q$ uniformly at random and computes $(G',M') = (G \cdot y^s, M \cdot g^s)$. We note that this does not require the knowledge of the secret key x.

Threshold ElGamal cipher Suppose the ElGamal secret key is generated using a distributed key generation protocol such as in [46] or [26]. We denote that x_j is the secret key of decryption party P_j and $y_j = g^{x_j}$ is the corresponding public key share. A quorum Q decryption parties working together can jointly reconstruct the ElGamal secret key as $x = \sum_{j \in Q} x_j \cdot \lambda_j$, where λ_j is the Lagrange coefficient for the jth share. To decrypt an ElGamal ciphertext $(G,M) = (my^r, g^r)$, the decryption party P_j first publishes M^{x_j}, and then generates a Chaum-Pedersen proof [13] that $log_g y_j = log_M M^{x_j}$ to prove that she has published the correct M^{x_j}. Finally, the plaintext m can be retrieved as $m = G/\prod_{j \in Q} M^{x_j \cdot \lambda_j}$.

Proxy re-encryption A proxy re-encryption [33] is a function to transfer an ElGamal encryption from one encryption key to another encryption key. Let (G_1, M_1) be an ElGamal encryption of a plaintext m using public key y_1, and let x_1 be the corresponding secret key, which is shared among a number of decryption parties using

a threshold scheme. A quorum Q of these parties can transfer (G_1, M_1) to an ElGamal encryption (G_2, M_2), which contains the same plaintext w.r.t. the public key y_2, without revealing m. Firstly, P_j selects a random value δ_j uniformly at random from Z_q, and computes $(\alpha_j, \beta_j) = (M_1^{-x_{1j} \cdot \lambda_j} \cdot y_2^{\delta_j}, g^{\delta_j})$. Then (G_2, M_2) can be computed as $(G_2, M_2) = (G_1 \cdot \prod_{j \in Q} \alpha_j, \prod_{j \in Q} \beta_j)$.

5.3 Proposed voting protocol

In the proposal scheme, a ballot form example is shown as in Figure 8. The candidate list is in the alphabetical order. We denote each row of the ballot form as a *ballot slice*. For each slice, the candidate name is printed on the left hand column and the right hand column is covered by some scratch surface.

Figure 8: A ballot form example

Stage 1. System setup

A number of tellers generate an ElGamal secret key $x_1 \in Z_q$ using a distributed key generation protocol [46, 26]. The corresponding public key $y_1 = g^{x_1}$ is made public. The voting machine generates another ElGamal private key $x_2 \in Z_q$ and publishes its public key $y_2 = g^{x_2}$. Furthermore, every clerk generates a private key for signing and reveals the corresponding public key.

Stage 2. Ballot construction

We first introduce how to generate a ballot slice, then it will be clear how to construct an entire ballot form.

Generating the receipt onions. The first step is for a set of l clerks to generate a large number of receipt onions. For each receipt onion, the jth clerk C_j randomly chooses a value $m_j \in G_q$ and a blinding factor $r_j \in Z_q$, and computes an ElGamal ciphertext

$$(G_{1j}, M_{1j}) = (m_j {y_1}^{r_j}, g^{r_j})$$

Then the receipt onion can be constructed in a public manner as

$$(G_1, M_1) = (\prod_{j=1}^{l} G_{1j}, \prod_{j=1}^{l} M_{1j}) = (m {y_1}^{r}, g^{r})$$

where $m = \prod_{j=1}^{l} m_j$ and $r = \sum_{j=1}^{l} r_j \pmod q$. All clerks who have been involved in the above procedure need to sign the receipt onion. The receipt onion and its signature $\{(G_1, M_1), \sigma\}$ are printed in the right hand column of a ballot slice.

Transforming receipt onions into ready onions When a large number of such ballot slices have been generated, several re-encryption parties will re-encrypt each of the receipt onions. This procedure is similar to PAV-Paillier. A re-encryption party receives a batch of ballot slices, she re-encrypts each onion in the right hand column, covers it by a scratch strip and overwrites it with its re-encrypted value. Then this party shuffles all the ballot slices and sends them to the next party. At this moment, there is a receipt onion in each ballot slice, but it is covered by multiple scratch strips. The only visible value is the ready onion on top of the scratch surface, which is the re-encryption of the receipt onion.

Transforming ready onions into booth onions For each ballot slice, a quorum of tellers apply the proxy re-encryption to transfer the ready onion which is encrypted under the public key y_1 into the booth onion which is encrypted under the public key y_2. The booth onion is printed in the left hand column. When this procedure finishes, the quorum of tellers use another scratch layer to cover the ready onion.

Printing the candidate name For each ballot slice, the voting machine decrypts the booth onion and overwrites it with the corresponding candidate name. Note that since the voting machine has the knowledge of the private key x_2, it can decrypt a booth onion $(G_2, M_2) = (m {y_2}^{t}, g^{t})$ and retrieve its plaintext m. Then the candidate name is derived as $h(m) \bmod v$, where h is a collision-resistant hash function and v is the number of candidates.

Construction of the entire ballot forms When generating a ballot slice, all parties know that they are generating a ballot slice for some candidate, but not for which candidate. Once a large number of such ballot slices have been generated, there should be some ballot slices for every candidate. We now sort these ballot slices into piles according to the candidate names. Therefore, an entire ballot form can be constructed by selecting a ballot slice from each pile and stick them together with the candidate in the alphabetical order. Therefore, a ballot form will be shown as in Figure 8.

Step 3. Check the ballot construction

In the voting booth, each voter will be provided with two ballot forms. The voter can randomly choose one

ballot form to challenge and casts her vote use the other one. To check whether a ballot form is correctly constructed, the voter removes the scratch surface in the right hand column. Then the voter requires the threshold tellers to decrypt one or several of the receipt onions, and checks whether the feedback of each receipt onion is the same as the candidate name printed in the left hand column.

Step 4. Ballot casting and tabulation

Our proposal scheme handles both approval elections and ranked elections. We will introduce how to cast a vote and how the received votes will be tallied in each case respectively.

Approval elections In the voting booth, each voter can randomly select a ballot form as shown in Figure 8. If a voter wishes to vote for Bob, she tears off the right hand column against Bob, as shown in Figure 9, destroys the remaining ballot form and submits the small piece to the election officials. The scratch surface is now checked, and only a vote with an intact scratch surface will be accepted. Then the voter removes the scratch surface in the presence of the election officials, revealing the receipt onion and its signature. The election officials check the signature, and return the small piece to the voter as the receipt after it has been scanned and signed. In the ballot tallying phase, all these received receipt onions will be tabulated by some mixnets, e.g. Neff's mix [41]. Finally, the shuffled votes will be decrypted by a quorum of tellers in a threshold fashion and the result will be announced.

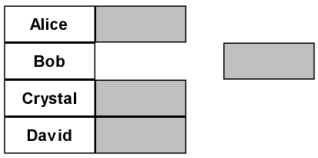

Figure 9: Cast a FPTP vote

Ranked elections We will introduce how to implement the proposal scheme in STV elections. Furthermore, it would be possible to implement it in some other ranked elections, such as Borda Court elections and Condorcet elections, if the ballot tallying phase has been modified according to the election requirement.

In STV elections, besides the ballot form, each voter will be provided with a voting card. A voter casts her vote as shown in Figure 10. She first tears off the right hand column against her most favourite candidate and sticks it to the first choice position in the voting card. Then she tears off the right hand column against her second preferred candidate and sticks it to the second choice position, and so on. The procedure of ballot submitting is similar: the voter removes the scratch surface in the presence of the election officials. Then the signatures are checked and the voting card will be returned to the voter as the receipt after it has been scanned and signed.

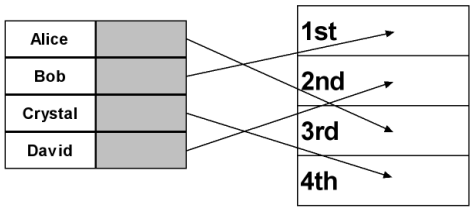

Figure 10: Cast an STV vote

To tally STV votes, we apply the techniques which have been introduced in [29]. For simplicity, the notation θ will be used to denote the receipt onions. For example, a receipt onion for Bob will be denoted as θ_B. At first, each received vote will be sorted in a public manner. E.g. the example vote will be sorted as:

$$(\theta_B, \theta_D, \theta_A, \theta_C)$$

As follows, all valid votes will be shuffled by mixnets to break the voter-vote relationship. The shuffle phase can be denoted as:

$$(\theta_B, \theta_D, \theta_A, \theta_C) \rightarrow (\theta'_B, \theta'_D, \theta'_A, \theta'_C)$$

Then each vote will be on-demand decrypted by a quorum of tellers in a threshold fashion. In the first round, only the first choice of each vote will be decrypted:

$$(Bob, \theta'_D, \theta'_A, \theta'_C)$$

If no candidate achieves the winning quota, some votes (from the lowest placed candidate) will be decrypted to reveal the second choice. This is done by removing the first choice to the end and treating the second choice as the first choice:

$$(David, \theta''_A, \theta''_C, \theta''_B)$$

Note that the transfer history of a vote is concealed, so the proposal scheme does not suffer the *Italian attack* in STV elections. Furthermore, if the election allows partial ranking, the receipt might reveal how many candidate a voter has marked. This drawback has been resolved in [29], and the same techniques can be applied here. But we will omit the details. For more information, see [29].

6 Discussion and Conclusion

In this paper, we have introduced an information leakage model to analyse information leakage in mixnet based e-voting schemes. The model might be found useful because of the following two properties:

(1). In some complex e-voting schemes, just by experience, it might be difficult to work out all possible threats against voter privacy. The information leakage model can help to list all these threats in a systematic manner[9]. Furthermore, different threats can be assigned to different experts, therefore it is possible to implement the threat analysis in parallel.

(2). In some circumstances, although it might be impossible to prevent all threats against voter privacy, the information leakage model can help to identify which parties could collude to obtain informaiton leakage. Therefore, we can select these parties from different groups with different interests. For example, in the PAV-Paillier scheme, because of the use of multiple layer of scratch surface, the last re-encryption party colluding with the voting machine might find out how a certain voter has cast her vote. Therefore, they should be carefully chosen so that they should not collude.

We have applied the information leakage model with the PAV-Paillier scheme, and our result shows that the scheme still suffers some threats which might leak voters' choice information. However, the majority of the threats can be mitigated as long as the scheme is carefully implemented. And among all existing e-voting schemes, we still consider PAV-Paillier to be one of the best e-voting schemes to provide voter privacy.

We also have introduced a variant of the PAV-Paillier scheme. We inherit similar ideas to construct the ballot forms, but Paillier encryption has been replaced by ElGamal encryption. This directly yields a more simple and straightforward threshold cryptosystem. Furthermore, it provides some other interesting properties such as the candidate list can be given in alphabetical order, the scheme can handle not only approval elections but also ranked elections, and it mitigates the randomisation attack.

Peter Ryan has suggested that our proposal would be very convenient for French elections, in which there is a separate ballot for each candidate. The voters will find all the different ballots in the polling station. Each voter has to pick several ballots before entering the voting booth and then puts the ballot for the preferred candidate in an envelope. Therefore, it would be not necessary to stick the ballot slices together.

[9]Note that we do not mean to find out all the detailed attacks. The discovery of these attacks still need experts' experience. But the information leakage model might help the experts to carry out the threat analysis in a logical manner.

7 Acknowledgement

The authors would like to thank Peter Ryan for many helpful discussions and suggestions. We have also had useful discussions with David Lundin, Kieran Leach and Roberto Araújo. Furthermore, we would like to thank the anonymous reviewers for their comments on the paper.

References

[1] ABE, M., AND IMAI, H. Flaws in some robust optimistic mixnets. *Proceedings of ACISP'03* (2003), 39–50. LNCS 2727.

[2] ADIDA, B. *Advances in cryptographic voting systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 2006.

[3] ADIDA, B., AND NEFF, C. A. Ballot casting assurance. *Proceedings of the 1st Usenix/ACCURATE Electronic Voting Technology Workshop* (2006). Vancouver, Canada.

[4] ADIDA, B., AND RIVEST, R. Scratch & Vote: self-contained paper-based cryptographic voting. *Proceedings of the 5th ACM workshop on Privacy in electronic society* (2006), 29–40.

[5] ADITYA, R. *Secure electronic voting with flexible ballot structure*. PhD thesis, Information Security Institute, Faculty of Information Technology, Queensland University of Technology, Australia, November 2005.

[6] BAUDRON, O., FOUQUE, P.-A., POINTCHEVAL, D., STERN, J., AND POUPARD, G. Practical multi-candidate election system. *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC'01)* (2001), 274–283. New York, NY, USA.

[7] BENALOH, J. Secret sharing homomorphisms: keeping shares of a secret secret. *Advances of Cryptology-Crypto'86* (1986), 251–260. LNCS 263.

[8] BENALOH, J. Towards simple verifiable elections. *Proceedings of Workshop on Trustworthy Election (WOTE'06)* (2006), 61–68. Cambridge, UK.

[9] BENALOH, J., AND TUINSTRA, D. Receipt-free secret-ballot elections (extended abstract). *Proceedings of the 26th Symposium on Theory of Computing (STOC'94)* (1994), 544–553. New York, NY, USA.

[10] BENALOH, J., AND YUNG, M. Distributing the power of a government to enhance the privacy of voters. *Proceedings of the 5th Symposium on Principles of Distributed Computing (PODC'86)* (1986), 52–62. New York, NY, USA.

[11] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[12] CHAUM, D. Secret ballot receipts: true voter-verifiable elections. *IEEE: Security and Privacy Magazine 2*, 1 (2004), 38–47.

[13] CHAUM, D., AND PEDERSEN, T. P. Wallet databases with observers. *Advances of Cryptology-Crypto'92* (1992), 89–105. LNCS 740.

[14] CHAUM, D., RYAN, P. Y. A., AND SCHNEIDER, S. A practical voter-verifiable election scheme. *Proceedings of the 10th European Symposium on Research in Computer Science (ESORICS'05)* (2005), 118–139. LNCS 3679.

[15] COHEN, J., AND FISHER, M. A robust and verifiable cryptographically secure election scheme. *Proceedings of the 26th IEEE symposium on the Foundations of Computer Science (FOCS'85)* (1985), 372–382.

[16] CRAMER, R., FRANKLIN, M., SCHOENMAKERS, B., AND YUNG, M. Multi-authority secret-ballot elections with linear work. *Advances of Eurocrypt'96* (1996), 72–82. LNCS 1070.

[17] CRAMER, R., GENNARO, R., AND SCHOENMAKERS, B. A secure and optimally efficient multi-authority election scheme. *Advances of Eurocrypt'97* (1997), 103–118. LNCS 1233.

[18] DAMGÅRD, I., AND JURIK, M. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. *Proceedings of PKC'01* (2001). LNCS 1992.

[19] DESMEDT, Y., AND KUROSAWA, K. How to break a practical MIX and design a new one. *Advances of Eurocrypt'00* (2000), 557–572. LNCS 1807.

[20] DOLEV, D., DWORK, C., AND NAOR, M. Non-malleable cryptography. *Proceedings of STOC'91* (1991), 542–552.

[21] ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on IT 31*, 4 (1985), 467–472.

[22] FIAT, A., AND SHAMIR, A. How to prove yourself: practical solutions to identification and signature problems. *Advances of Cryptology-Crypto'86* (1986), 186–199. LNCS 263.

[23] FOULLE, S., SCHNEIDER, S., TRAORÉ, J., AND XIA, Z. Threat analysis of a practical voting scheme with receipts. *Proceedings of the 1st International Conference of E-Voting and Identity (VOTE-ID 2007)* (2007), 156–165. LNCS 4896.

[24] FURUKAWA, J., MIYAUCHI, H., MORI, K., OBANA, S., AND SAKO, K. An implementation of a universally verifiable electronic voting scheme based on shuffling. *Proceedings of Financial Cryptography (FC'02)* (2002), 16–30. LNCS 2357.

[25] FURUKAWA, J., AND SAKO, K. An efficient scheme for proving a shuffle. *Advances of Cryptology-Crypto'01* (2001), 368–387. LNCS 2139.

[26] GENNARO, R., JARECKI, S., KRAWCZYK, H., AND RABIN, T. Secure distributed key generation for discrete-log based cryptosystems. *Advances of Eurocrypt'99* (1999), 295–310. LNCS 1592.

[27] GOLLE, P., ZHONG, S., BONEH, D., JAKOBSSON, M., AND JUELS, A. Optimistic mixing for exit-polls. *Advances of Asiacrypt'02* (2002), 451–465. LNCS 2501.

[28] GROTH, J. A verifiable secret shuffle of homomorphic encryptions. *Proceedings of PKC'03* (2003), 145–160. LNCS 2567.

[29] HEATHER, J. Implementing STV securely in Prêt à Voter. *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF'07)* (2007), 157–169. Venice, Italy.

[30] HIRT, M., AND SAKO, K. Efficient receipt-free voting based on homomorphic encryption. *Advances of Eurocrypt'00* (2000), 539–556. LNCS 1807.

[31] JAKOBSSON, M. A practical mix. *Advances of Eurocrypt'98* (1998), 449–461. LNCS 1403.

[32] JAKOBSSON, M. Flash mixing. *Proceedings of the 18th annual ACM symposium on Principles of Distributed Computing (PODC'98)* (1999), 83–89.

[33] JAKOBSSON, M. On quorum controlled asymmetric proxy reencryption. *Proceedings of PKC'99* (1999), 112–121. LNCS 1560.

[34] JAKOBSSON, M., JUELS, A., AND RIVEST, R. L. Making mix nets robust for electronic voting by randomized partial checking. *Proceedings of the 11th USENIX Security Symposium* (2002), 339–353.

[35] JAKOBSSON, M., SAKO, K., AND IMPAGLIAZZO, R. Designated verifier proofs and their applications. *Advances of Eurocrypt'96* (1996), 143–154. LNCS 1070.

[36] KARLOF, C., SASTRY, N., AND WAGNER, D. Cryptographic voting protocols: a systems perspective. *Proceeding of USENIX Security Symposium* (2005), 186–200. LNCS 3444.

[37] LEE, B., AND KIM, K. Receipt-free electronic voting through collaboration of voter and honest verifier. *Proceedings of JW–ISC 2000* (2000), 101–108. Okinawa, Japan.

[38] MAGKOS, E., BURMESTER, M., AND CHRISSIKOPOULOS, V. Receipt-freeness in large-scale elections without untappable channel. *The 1st IFIP Conference on E-commerce/E-business/E-government* (2001), 683–693. Zurich.

[39] MITOMO, M., AND KUROSAWA, K. Attack for flash mix. *Advances of Asiacrypt'00* (2000), 192–204. LNCS 1976.

[40] MORAN, T., AND NAOR, M. Receipt-free universally-verifiable voting with everlasting privacy. *Advances of Cryptology-Crypto'06* (2006), 373–392. LNCS 4117.

[41] NEFF, C. A. A verifiable secret shuffle and its application to e-voting. *Proceedings of the 8th ACM conference on Computer and Communications Security (CSS'01)* (2001), 116–125.

[42] NEFF, C. A. Practical high certainly intent verification for encrypted votes. *VoteHere document* (2004).

[43] NGUYEN, L., SAFAVI-NAINI, R., AND KUROSAWA, K. Verifiable shuffles: a formal model and a Paillier-based efficient construction with provable security. *Proceedings of ACNS'04* (2004), 61–75. LNCS 3089.

[44] PAILLIER, P. Public-key cryptosystems based on discrete logarithms residues. *Advances of Eurocrypt'99* (1999), 223–238. LNCS 1592.

[45] PARK, C., ITOH, K., AND KUROSAWA, K. Efficient anonymous channel and all/nothing election scheme. *Advances of Eurocrypt'93* (1993), 248–259. LNCS 756.

[46] PEDERSEN, T. A threshold cryptosystem without a trusted party. *Advances of Eurocrypt'91* (1991), 522–526. LNCS 547.

[47] PFITZMANN, B. Breaking an efficient anonymous channel. *Advances of Eurocrypt'94* (1994), 339–348. LNCS 950.

[48] RYAN, P. Y. A. Prêt à Voter with Paillier encryption. *Technical Report of University of Newcastle CS-TR:1014* (2007).

[49] RYAN, P. Y. A. Prêt à Voter with Paillier encryption. *Journal of Mathematical Modelling of Voting Systems and Elections: Theory and Applications, special issue of Mathematics and Computer Modelling, Ed Alexander S. Belenky* (2008).

[50] RYAN, P. Y. A., AND PEACOCK, T. Prêt à Voter: a system perspective. *Technical Report of University of Newcastle CS-TR:929* (2005).

[51] RYAN, P. Y. A., AND SCHNEIDER, S. Prêt à Voter with re-encryption mixes. *Proceedings of the 11th European Symposium on Research in Computer Science (ESORICS'06)* (2006), 313–326. LNCS 4189.

[52] SAKO, K., AND KILIAN, J. Receipt-free mix-type voting scheme. *Advances of Eurocrypt'95* (1995), 393–403. LNCS 921.

[53] SCHNORR, C. P. Efficient signature generation by smart cards. *Journal of Cryptology* (1991), 161–174.

[54] SHOUP, V. Practical threshold signature. *Advances of Eurocrypt'00* (2000), 207–220. LNCS 1807.

[55] WIKSTRÖM, D. Five practical attacks for optimistic mixing for exit-polls. *Proceedings of SAC'03* (2003), 160–175. LNCS 3006.

[56] XIA, Z., SCHNEIDER, S., HEATHER, J., RYAN, P. Y. A., LUNDIN, D., PEEL, R., AND HOWARD, P. Prêt à Voter: All-In-One. *Proceedings of Workshop On Trustworthy Elections (WOTE 2007)* (2007), 47–56. Ottawa, Canada.