Receipt Freeness of Prêt à Voter Provably Secure

Dalia Khader[1] and Peter Y. A. Ryan[1,2]

[1] Interdisciplinary Centre for Security Reliability and Trust, SnT
[2] Université du Luxembourg

Abstract. Prêt à Voter is an end-to-end verifiable voting scheme that is also receipt free. Formal method analysis was used to prove that Prêt à Voter is receipt free. In this paper we use one of the latest versions of Prêt à Voter [XCH+10] to prove receipt freeness of the scheme using computational methods. We use provable security game models for the first time to prove a paper based voting scheme receipt free. In this paper we propose a game model that defines receipt freeness. We show that in order to simulate the game we require IND-CCA2 encryption scheme to create the ballots. The usual schemes used in constructing Prêt à Voter are either exponential ElGamal or Paillier because of their homomorphic properties that are needed for tallying, however both are IND-CPA secure. We propose a new verifiable shuffle "D-shuffle" to be used together with an IND-CPA encryption schemes that guarantees that the outputs of the shuffle are IND-CCA2 secure ciphertexts and they are used for constructing the ballots. The idea is based on Naor-Yung transformation [NY95]. We prove that if there exist an adversary that breaks receipt freeness then there exist an adversary that breaks the IND-CCA2 security of Naor-Yung encryption scheme. We further show that the "D-Shuffle" provides us with the option of having multiple authorities creating the ballots such that no single authority can break voter's privacy.

1 Introduction

For centuries, security properties of an election were maintained via enforcing physical constraints of the real-world. For example, an isolated booth and a locked ballot box, provided privacy of the vote and ensured coercion-resistance and fairness; Transparency of the procedure where anyone can view the tally, casting, counting, etc, was a form of verifiability of the procedure, and a guarantee of correctness, and robustness. Replicating these security notions in the digital setting is what we refer to as secure electronic voting (E-voting).

Research in e-voting can be divided into two schools of thought. The first was to have remote voting, in which voters submit their votes via some electronic mean that is connected to some network (maybe even WWW) and gathered via some tallying authority(s) with a public bulletin board [RT09,AMPQ09]. The second school of thought was to use paper-based e-voting where paper ballots have certain cryptographic elements that enable secrecy and the tallying is done with the help of machines [XSH$^+$07,FCS06]. The idea is to allow verifiable yet confidential voting. Even though both research lines use cryptography to ensure the security properties required, hardly any work in the literature managed to prove these properties secure under computational models.

The main security property that concerned researchers when designing an e-voting system was privacy of the vote. *Privacy* implies voters' vote is kept confidential [BHM08,DKR09,MN06]. The following are different terminologies used in literature to express privacy:

– Ballot secrecy. A voter's vote is not revealed to anyone and the vote is computationally hidden using cryptographic elements.
– Receipt freeness. A voter cannot gain information which can be used to prove, to a coercer, how she voted.
– Coercion resistance. A voter cannot collaborate, with a coercer, to gain information which can be used to prove how she voted.
– Ever lasting privacy. The link between the voter and her vote is destroyed completely such that no computational power be it now or in the future can retrieve it.

Ideally, ever lasting privacy seems to be the ultimate goal and there are schemes in the literature that achieve it [MN06,MN10] but on the cost of verifiability, unconditional integrity or by introducing other assumptions. However, a lot of researchers prefer to use schemes that compromise the level of privacy to gain either usability properties of the voting scheme or verifiability properties. Verifiability is a notion that includes three aspects [JCJ05,Dag07,KRS10].

– Individual verifiability. A voter can check that her own encrypted ballot is published on the election's bulletin board and that her vote is correctly encoded in the ballot.
– Universal verifiability. Anyone can check that all the votes in the election outcome correspond to ballots published on the election's bulletin board.
– Eligibility verifiability. Anyone can check that each ballot published on the bulletin board was cast by a registered voter and at most one ballot is tallied per voter.

Even though, e-voting schemes have deployed cryptography to insure privacy on any level, hardly any computational models were used in proving their security. The only attempts that we are aware of in the literature was Moran and Naor 's work [MN06,MN10] who proposed an ever lasting privacy scheme and used UC models to prove security. Lately, Bernhard et al. [BCP$^+$11] proposed a voting friendly encryption scheme that can be used in Helious voting system and proved that such a scheme is needed to prove ballot secrecy using provable security techniques.

In this paper, we have chosen one of the well known paper based e-voting schemes, Prêt à Voter , and proved its receipt freeness using provable security game models. This is the first paper-based voting scheme proven secure receipt free using a computational model. A few formal methods techniques were used earlier on to prove Prêt à Voter secure [XSHT08,RBH$^+$09]. The scheme achieves end-to-end verifiability and receipt freeness.

2 Preliminaries

This section presents the assumptions and cryptographic primitives that will be used throughout the paper. We shall start with some notations and conventions. Let $\mathcal{H}$ denote a hash function and (p, q, g) be cryptographic parameters, where p and q are large primes such that $q \mid p-1$ and g is a generator of the multiplicative subgroup $\mathbb{Z}_p^*$ of order q.

2.1 Baudron's Homomorphic Counter [BFP$^+$01]

Suppose there are $(k+1)$ candidates and the total number of eligible voters is n, a value L is chosen such that $n < 2L$. 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. Homomorphically adding the encrypted values of election parameter will end up with a ciphertext $CT = Enc(v)$ where $v = 2^0 c_0 + 2^L c_1 + 2^{2L} c_2 \cdots + 2^{kL} c_k$, where c_j is the number of votes that went for candidate j for any $j \in \{0, \ldots, k\}$. To implement such a counter, Paillier and Exponential ElGamal[1] can be used.

2.2 Verifiable Shuffles

A shuffle is a permutation and re-randomization of a set of ciphertexts. A shuffle, given n ciphertexts as an input $\{CT_i\}$ output n ciphertexts $\{\acute{CT_i}\}$. Shuffling itself is easy, the challenge is to provide a proof of correctness of a shuffle that anyone can verify. Verifiable shuffles have been used widely in voting schemes. The main motivation behind using them, is to submit encrypted votes into some mix-net where every mix-server shuffles the votes. The output of the mix-net is then decrypted, allowing anonymity of the voters to be maintained. Using verifiable shuffles stops mix-servers from cheating. Verifiable shuffles in the early studies required a lot of computational overhead [SK95,Abe99]. Neff [Nef01] suggested using zero knowledge proofs for the correctness of ElGamal ciphertext shuffles. The proofs were based on the invariance of polynomials under permutation of the roots. They were a 7-move proof. The number of rounds dropped to 3-moves in [FS02,FMOS02] where the general idea was to commit to a permutation matrix and prove that the ciphertexts have been shuffled accordingly. As in Neff's work, the schemes were based on ElGamal encryption.

[1] The value v can be efficiently computed (the maximum value is if all voters vote for the last candidate) using baby-step giant-step algorithm (this is possible because the values of k tend to be small), and $c_1, \ldots, c_k$ can be recovered using the super-increasing nature of the encoding and with the help of algorithms such as the knapsnack algorithm.

However similar techniques were used in Paillier encryption too [NSNK05,OT04]. Some verifiable shuffles were based on homomorphic integer commitments as the building block [Wik05a,Wik05b,GL07]. Other shuffles were based on cut/choose techniques, permutation networks,etc [Abe99,Wik05a,Wik05b,MA99]

3 Prêt à Voter

Prêt à Voter was first proposed by Ryan in [Rya05]. It is a paper based electronic voting scheme that is known to be receipt free. In this section we describe the idea behind the design of the scheme and we also propose a game model for receipt freeness.

3.1 Prêt à Voter Overview

To explain how Prêt à Voter works, we need to explain the structure of the ballots. Prêt à Voter is based on the assumption that the voter is given her ballot confidentially. The ballot is divided into two sides: the left hand side (LHS) which has the list of candidates name permuted randomly, and the right hand side the onion. The onion is an encryption of the order of the candidates in the LHS named as the onion for historical reasons. Each ballot has a unique serial number, SN, for administrative purposes such as searching for the ballot on the bulletin board, etc (See Figure 1).

The voting ceremony takes place in the booth. The voter takes her ballot in the booth, ticks next to the name of the candidate she wants to vote for. She separates the RHS from LHS. She shreds the LHS and comes out with the RHS. She takes the remaining part of the ballot to the polling station's employee who will scan it, send it to the tallying authority and give a signed copy to the voter to keep. The scanned information is enough to help in tallying the result because the authorities responsible of the count can decrypt the onions to know the candidate corresponding to the choice of the voter. The voter can verify that their votes have been received by comparing the onion, serial number and index of choice to what is announced on the bulletin board. The details of the procedure of tabulation, randomization of ballots, tallying, distributing the ballots, etc, varies in the different versions of Prêt à Voter [Rya05,RBH$^+$09,XCH$^+$10]. On a conceptual level the procedure is the same. Auditing the ballots is an idea that also has been considered in all versions of Prêt à Voter in literature but differed slightly in style. Informally, the auditing procedure is revolving around decrypting a number of onions and checking that they correspond to the LHS order. Given that the authorities responsible of creating the ballots can not predict which ballots will be chosen for auditing, it is hard to cheat without a high possibility of getting caught.

3.2 Prêt à Voter Receipt Freeness Game model

Given we intend to use provable security techniques in proving receipt freeness, we should define the term "Receipt Free" using a game model. We propose

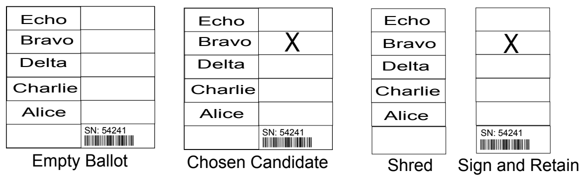

Fig. 1. Prêt à Voter : The Ceremony

a game between a hypothetical adversary $\mathcal{A}$ and challenger $\mathcal{C}$. $\mathcal{C}$ runs certain oracles that $\mathcal{A}$ can query. We first introduce the following oracles:

– **Retrieve Empty Ballot (REB):** $\mathcal{A}$ sends a serial number to $\mathcal{C}$ of which ballot he would like to get. $\mathcal{C}$ responds with an empty ballot (i.e. both RHS and LHS).
– **Retrieve Half Ballot (RHB):** $\mathcal{A}$ sends a serial number to $\mathcal{C}$ who responds with the RHS of the ballot, unfilled.
– **Retrieve Used Ballot (RUB):** $\mathcal{A}$ sends a serial number and index to $\mathcal{C}$, who responds with the RHS of the ballot ticked on that index (i.e returns the receipt of the ballot). Furthermore, the bulletin board is updated since the ballot has been used and a vote should be counted.
– **Vote:** $\mathcal{A}$ sends a vote (i.e a RHS of ballot ticked on any position of $\mathcal{A}$'s choice). The bulletin board is updated.
– **Reveal Candidate Order (RCO):** $\mathcal{A}$ sends the RHS of the ballot used or not, $\mathcal{C}$ responds with the candidate order on that ballot.
– **Create New Ballot (CNB):** $\mathcal{A}$ creates a ballot of his own and sends to $\mathcal{C}$. $\mathcal{C}$ saves the ballot in a database.
– **Reveal Status of Bulletin Board (RBB):** $\mathcal{A}$ sends a request to reveal status of the bulletin board. $\mathcal{C}$ returns the tallying result of the board until the moment of the query. Votes can be later added and $\mathcal{C}$ can query RBB again.
– **Reveal Partial Status of Bulletin Board (PRBB):** $\mathcal{A}$ sends a set of used ballots. $\mathcal{C}$ responds with the partial tallying result of that set.

Giving the adversary access to these oracles allows us to assume the existence of a powerful attacker and proving receipt freeness under such assumption. It is worth noting that such oracles can allow the adversary to conduct other types of attacks but the game model we are defining is addressing receipt freeness only. We give $\mathcal{A}$ the ability to tally using RBB and PRBB. He can stuff ballots using CNB, he can corrupt votes using Vote oracle, and he can corrupt ballots using RCO. The first three oracles, (REB, RHB, RUB), help in enabling the adversary to obtain inputs from the system and observations. For example, $\mathcal{A}$ may use the ballots retrieved from RHB or REB to query oracles such as RCO or Vote.
The game model is designed with the assumption that we have two candidates only (Alice and Bob) for simplicity of explanation, however, it can be extended to include more. We say that a "Prêt à Voter " voting scheme is receipt free if

no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following RF-game:

- RF.Setup: $\mathcal{C}$ sets up the system by creating the private and public parameters, then starting with an empty bulletin board. The bulletin board is accessible to $\mathcal{A}$ as "read only". $\mathcal{C}$ sends the public parameters to $\mathcal{A}$. Any private parameters used in tallying and/or decrypting ballots are kept secret.
- RF.Phase[I]: $\mathcal{A}$ queries oracles REB, RHB, RUB, RCO, CNB, RBB, PRBB and Vote.
- RF.Challenge: $\mathcal{A}$ picks two ballots it would like to challenge. Assume the serial numbers of these ballots are i and j. The two ballots should have not been queried in oracles REB, RUB, Vote, and RCO. $\mathcal{C}$ returns the two receipts, one as a vote to Alice and the other as vote to Bob.
- RF.Phase[II]: $\mathcal{A}$ queries oracles REB, RHB, RUB, RCO, CNB, RBB, PRBB and Vote. Except he can not query the challenged ballots. He can though verify the challenge by querying RBB and PRBB.
- RF.Guess: The adversary returns a serial number $b \in \{i, j\}$ as its guess to which ballot was used in voting for Alice. If the guess is correct then $\mathcal{A}$ wins the game and the output is 1, if the adversary does not guess it right then the output is 0.

In this game model all trusted parties are considered one entity and that is the challenger $\mathcal{C}$ and all corrupted parties are considered one entity and that is the adversary $\mathcal{A}$. The game focuses only on defining receipt freeness, so we assume the adversary is not interested in other types of attack. As long as the ballots he is challenging are not corrupted, the adversary should not be able to distinguish between the receipt of i and the receipt of j. We refer to the game model above as EXP_{RF}, the probability of guessing the receipt in case of two candidates should be 50%. Receipt Freeness for a two candidate elections is defined in Definition 1.

Definition 1. *A "Prêt à Voter " voting scheme is Receipt Free if for all polynomial time adversaries $\mathcal{A}$, $Adv.RF(k) = |Pr(EXP_{RF} = 1) - 1/2| \leq \epsilon$ where k is the security parameter and ϵ is negligible.*

4 Construction of a new " Prêt à Voter "

In this work we tried avoiding creating another version of "Prêt à Voter ". We use the version published in [XCH^{+}10] as the construction we aim to prove secure. We soon realize that to simulate the oracles mentioned earlier one would require an IND-CCA2 secure encryption scheme to create the onions. The known homomorphic encryptions that can be used in voting are exponential ElGamal and Paillier, and both are IND-CPA secure. See Appendix B for definitions. Thanks to the work of Naor and Yung [NY95], a general construction was used to transform any IND-CPA secure encryption scheme to an IND-CCA secure encryption scheme. Later on, Sahai realized that such a construction achieves IND-CCA2 security [Sah99]. In this section we explain the Naor and Yung transformation.

We then show how one can add minimum modifications to "Prêt à Voter " introduced in [XCH$^+$10] in order to help simulate the game model above and prove receipt freeness. Throughout this paper we will refer to the scheme as [XCH$^+$10] and to the new version as PAV.

4.1 Construction of [XCH$^+$10]

The general idea of [XCH$^+$10] is to have each candidate coded with a Baudron's Homomorphic Counter (See Figure 2, part A). The codes encrypted are homorphically added as explained in §2.1. The onion is composed of two parts as follows:

- **Ciphertexts**: An encrypted list of the different candidates i.e. a permutation of $\{Enc(2^0, r_0)$, $Enc(2^L, r_1)$, $Enc(2^{2L}, r_2)$, $\dots$, $Enc(2^{kL}, r_k)\}$ that corresponds to the order of the candidates on the LHS and r_i is a randomization factor.
- **Proofs**: Non-interactive zero knowledge proofs to check that the ballot is well formed. That means proving that each part of the onion encrypts a unique counter.

There are two encryption schemes widely used for e-voting for their homomorphic properties: ElGamal and Paillier. In [XCH$^+$10] the authors chose Paillier and used verifiable shuffles to check that the ballot is well formed. To generate a ballot the authors suggest the following list as the first input to the first mix-server: $\{Enc(2^0, 1)$, $Enc(2^L, 1)$, $Enc(2^{2L}, 1)$, $\dots$, $Enc(2^{kL}, 1)\}$. Anyone can verify that the ciphertexts are unique encryptions of the counters since the randomization value is 1. Then the ciphertexts are re-encrypted and shuffled in a verifiable manner. The proofs of these shuffles are published. In [XCH$^+$10] do not specify which shuffle to use. We will recommend in this paper a verifiable shuffle (in §4.3) that helps in simulating the oracles in the receipt free game model. The shuffle is based on ElGamal encryption because we use the zero knowledge proofs in Appendix §A and therefore we assume that exponential ElGamal is used for Prêt à Voter .

4.2 Naor-Yung Transformation

The general idea proposed by Naor-Yung is to start with any IND-CPA encryption scheme E (See Appendix B). The new encryption scheme $NY.E$ of a message is two distinct encryptions under the original scheme and a simulation sound zero-knowledge proof that the two ciphertexts encrypt the same message. Let $E = (KeyGen, Enc, Dec)$ be a public key encryption scheme. Let $P = (Prove$, $Verify$, $Simulate)$ be a non-interactive zero knowledge proof of knowledge scheme for proving: $PoK\{(m, r_1, r_2) : c_1 = Enc(pk_1, m, r_1) \wedge c_2 = Enc(pk_2, m, r_2)\}$ with uniquely applicable proofs. Naor-Yung Transformation [NY95] encryption scheme $NY.E = (NY.KeyGen$, $NY.Enc$, $NY.Dec)$ is as follows:

– $NY.KeyGen(k)$: Takes security parameter k and runs the key generator of the original scheme twice to produce to pairs of keys $(sk_1, pk_1) = KeyGen(k)$ and $(sk_2, pk_2) = KeyGen(k)$. The secret key $NY.sk = (sk_1,\ sk_2)$ and public key $NY.pk = (pk_1, pk_2)$.
– $NY.Enc(m, NY.pk)$: Choose the randomization factors r_1, r_2, compute $c_1 = Enc(pk_1, m, r_1)$, $c_2 = Enc(pk_2, m, r_2)$, and $\pi = Prove(m,\ pk_1,\ pk_2,\ r_1,\ r_2,\ c_1,\ c_2)$. Ciphertext of the transformation would be $NY.CT = (c_1, c_2, \pi)$.
– $NY.Dec(c_1, c_2, \pi)$: If $Verify(c_1, c_2, \pi) = 1$ then $m = Enc(c_1, sk_1)$ else abort[2].

Theorem 1. ***(Sahai [Sah99]):*** *If the zero knowledge proof system P is a proof of knowledge and has uniquely applicable proofs and if the encryption scheme E is IND-CPA then applying Naor-Yung transformation gives an encryption scheme NY.E that is IND-CCA2 secure.*

We are going to use Theorem 1 to make Prêt à Voter provably secure under the game model defined in §3.2. The main goal is to simulate the game model of receipt freeness. If we have an IND-CPA secure encryption scheme we do not have access to a decryption oracle. The decryption oracle will help us with simulating oracles such as RCO,CNB,RBB,and PRBB (details in §5). We shall use Naor-Yung's transformation to get an IND-CCA2 scheme. However, our intention is to avoid designing a new version of Prêt à Voter for that purpose and use the one proposed by [XCH$^+$10]. We notice that in the original paper the authors use verifiable shuffle proofs as a proof that the ballots are well formed. They did not specify which shuffle in literature is most suitable. In this work, we propose a verifiable shuffle that can include the Naor-Yung's proof. The shuffle proposed is inspired from the designs in [MA99,HS00]. Section §4.3 explains the shuffle while §4.4 explains how to use it for Prêt à Voter ballots.

4.3 The D-Shuffle

The main aim of proposing the D-shuffle is to have a verifiable shuffle that has all inputted ciphertexts as IND-CCA2 secure and all outputted ciphertexts as IND-CCA2 secure. The latter requirement guarantees that once we use the outputs of a mix-net to create a ballot for Prêt à Voter , we get a ballot onion that is IND-CCA2 secure.

In literature, Wikström [Wik06] proposed a mix-net that uses the concept of augmented cryptosystems. Wikström's idea was to provide a mix-net that is secure under the universal composability model. He needed to provide a level of IND-CCA2 security in the mix-net yet maintain the homomorphic properties needed. Initially his solution seemed to address our problem. He uses a Cramer Shoup like encryption which is IND-CCA2 secure and extracts from it an El-Gamal encryption which is homomorphic. However, Wikström mix-net does not help our purpose because the last outputted results of the mix-net are not IND-CCA2 secure. They are pure ElGamal encryptions that are IND-CPA secure.

[2] One can also decrypt $m = Enc(c_2, sk_2)$.

Definition 2. *Sequence $A = (c_0, \ldots, c_k)$ is a superincreasing sequence if every element of the sequence is positive and is greater than the sum of all previous elements in the sequence (i.e. $c_n > \sum_{i=0}^{n-1} c_i$).*

Theorem 2. *Let $(c_0, \ldots, c_k)$ be a super-increasing sequence, $A = \{c_0, \ldots, c_k\}$, and $C = \sum_{i=0}^{k} c_i$. If $(x_0, \ldots, x_k)$ is a solution of*

$$C = \sum_{i=0}^{k} x_k$$

such that $\forall j \in \{0, \ldots, k\} : x_j \in A$, then $(x_0, \ldots, x_k)$ is a permutation of A.

Proof. Recall the subset sum problem; Given a set of integers A and an integer C, find any non-empty subset X that sums to C. The subset sum problem is proven to have either one unique solution or none [Nea87] over super-increasing sequences. Given Theorem 2 assumes the existence of the subset $X \subseteq A$ and assumes that $C = \sum\limits_{i=0}^{k} x_k$ then by the uniqueness property $X = A$.

The general idea behind our shuffle is derived from Theorem 2. Furthermore, the shuffle relies on two assumptions: the set of plaintexts are of a super-increasing property and are public knowledge. These two assumptions are reasonable if we are to use the shuffle for voting schemes such as Prêt à Voter that have the plaintexts as candidate codes described by Baudron's Homomorphic Counter §2.1. The following two conditions derived from Theorem 2 implies a verifiable and honest shuffle.

- Theorem 2 states $\forall i,\ i \in [0, k],\ x_i \in A$. In a verifiable shuffle this is equivalent to saying "All outputted ciphertexts belong to the list of all inputted ciphertexts". The 1-out-of-L re-encryption proof §A will be used for that purpose. Furthermore, the first inputted ciphertexts to the mix-net should encrypt values of super-increasing properties. Proving that the values are super-increasing is difficult but if the original plaintext is publicly known, i.e. $A = \{c_0, \ldots, c_k\}$ is known to everyone, then one can use the disjunctive proof of equality between discrete logs §A to prove that each ciphertext encrypts a plaintext from that set.
- Theorem 2 states that $C = \sum\limits_{i=0}^{k} c_i = \sum\limits_{i=0}^{k} x_k$. In a verifiable shuffle this is equivalent to saying that the homomorphic summation of the inputted ciphertexts and the homomorphic summation of the outputted ciphertexts are encryptions of the same plaintext value. Plaintext equivalence test is done using the discrete logarithm equality proofs §A.

The following are the steps defining the D-shuffle which include the Naor-Yung's tranformation and how that transformation is transfered from one mix-server to the other, maintaining the IND-CCA2 security of the outputted ciphertexts. Note that the index j is used for mix server, and i is an index for ciphertexts. That is, $CT_{i,j}$ is the i-th ciphertext inputted to the j-th mix-server.

– The initial ciphertexts present a permutation of $A = \{Enc(pk_1, c_0, rnd_{0,0}), Enc(pk_1, c_1, rnd_{0,1}), \ldots, Enc(pk_1, c_k, rnd_{0,k})\}$ for random values $\{rnd_{0,1}, \ldots, rnd_{0,k}\} \in \mathbb{Z}_p^*$. Furthermore, we have the encryption $B = Enc(pk_2, C, RND_0)$ where $C = \sum_{i=1}^{k} c_i$ and $RND_0 \in \mathbb{Z}_p^*$ is a randomization factor. A proof π is provided, of equality of plaintext between the encrypted C and the homomorphic sum of ciphertexts in A. Note that the elements above do form a Naor-Yung transformation. The last proof to be provided is a disjunctive proof of equality between discrete logs that shows the ciphertexts in A do encrypt the values $\{c_0, \ldots, c_k\}$.

– Each mix-server j, re-encrypts each inputted ciphertext i it received with random values $rnd_{i,j}$. It then permutes the ciphertexts and provides a 1-out-of-L re-encryption proof for elements of A, and a proof that if the re-encrypted values of ciphertexts in A are homomorphically summed, and compared to the re-encrypted value of B, the plaintext is still equivalent. This is done by proving that the same randomization factors were used for both as follows:

1. Let $\alpha_0 = \sum_{i=0}^{k} rnd_{0,i}$; Initial values received by the first mix-server are:

$$B_0 = Enc(pk_2, C, RND_0) = (u_0, v_0) = (g^{RND_0}, g^{RND_0 sk_2} g^C)$$

$$HB_0 = \prod_{i=0}^{k} Enc(pk_1, c_i, rnd_{0,i}) = (\bar{u_0}, \bar{v_0}) = (g^{\alpha_0}, g^{sk_1 \alpha_0} g^C)$$

2. Re-encryption after mix-server j chooses $\{rnd_{j,0}, \ldots, rnd_{j,k}\}$:

Let $RND_j = \sum_{i=0}^{k} rnd_{j,i}$ and $\alpha_j = \sum_{i=0}^{k} \sum_{l=0}^{j} rnd_{l,i} = RND_j + \alpha_{j-1}$,

$$B_j = ReEnc(B_{j-1}, RND_j) =$$

$$(u_j, v_j) = (g^{(RND_j + \sum_{l=0}^{j-1} RND_l)}, g^{(RND_j + \sum_{l=0}^{j-1} RND_l) sk_2} g^C)$$

$$HB_j = \prod_{i=0}^{k} ReEnc(CT_{i,j-1}, rnd_{j,i}) =$$

$$(\bar{u_j}, \bar{v_j}) = (g^{(\alpha_{j-1} + RND_j)}, g^{(\alpha_{j-1} + RND_j) sk_1} g^C)$$

3. Provide a proof of equality between discrete logs of all three elements below §A:

$$\frac{u_j}{u_{j-1}} = \frac{\bar{u_j}}{\bar{u_{j-1}}} = g^{RND_j} \; ; \; \frac{v_j}{v_{j-1}} = g^{RND_j sk_2} = (pk_2)^{RND_j} \; ;$$

$$\frac{\bar{v_j}}{\bar{v_{j-1}}} = g^{RND_j sk_1} = (pk_1)^{RND_j}$$

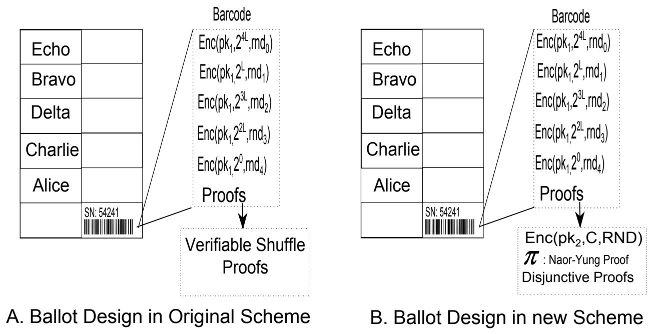

Fig. 2. Prêt à Voter : The Ballot Design

The shuffle is sound, and correct given Theorem 2. The shuffle is verifiable given the proofs of knowledge. The shuffle maintains anonymity and unlinkability given the zero knowledge property of the proofs of knowledge. Finally, the encrypted ciphertexts of the shuffle have to be IND-CCA2 secure given that Naor-Yung transformation is maintained.

4.4 The new Construction of PAV

In Prêt à Voter schemes shuffles are used for creating the ballots. The candidates on each ballot are shuffled using a mix-net. Research has been done on Prêt à Voter to create the ballots in a distributed manner [RS06]. However, in [XCH$^+$10], the authors point out that achieving distributed ballot generation is not easy in their scheme for three reasons: [I] Proving the ballot is well-formed in the distributed fashion is not easy. [II] Printing the ballot without the printer(s) learning the candidates order is difficult. [III] How to ensure robustness so that the scheme can be run even in the presence of some dishonest election officials.

In this paper we provide two solutions; One improves security and allows a distributed way of creating the ballot making use of properties of the D-shuffle. However, usability of the scheme maybe compromised and needs to be further studied. On the other hand, we also propose a solution with the same assumption as done in [XCH$^+$10] of one authority but keep the desirable property of Prêt à Voter being easy to use.

Single Authority: One can use a mix-net that has one mix-server. If we are to use the D-Shuffle then one can use the initial values B_0, HB_0, the Naor-Yung proof π, and the disjunctive proofs that plaintexts belong to Baudron's Homomorphic Counter to present the proofs in the onion (See Figure 2) together with the ciphertexts of the Baudron's Counters. Figure 2 shows the ballots of the original scheme [XCH$^+$10] in part 'A' and the proposed change in 'B'.

Multi-Authorities: Assume we use the bulletin board to publish all zero knowl-

edge proofs used in the shuffle. The bulletin board is secure in the sense it maintains history and ballot generation authorities, can only add information to it. This allows public verifiability of the shuffles.
We plan to use the proposed D-Shuffle for creating the ballots and offer a print-on-demand option to voters as done in [RS06]. In this case we have two onions (RHS.onion and LHS.onion) and no candidate names. On the day of voting, the voter can ask for the LHS to be generated and printed. The D-shuffle is used to randomize the RHS and LHS as ciphertexts while providing zero knowledge proofs to show that the sides correspond to each other. Assume we have ElGamal encryption for the LHS with public keys $(LHS.pk, LHS.sk)$ and one for the RHS $(RHS.pk, RHS.sk)$ where $RHS.E$ is a Naor-Yung transformation. The following describes the ballot generation using the D-shuffle:

- The RHS is done as described in the D-shuffle with the initial ciphertexts being a permutation of $A_{RHS} = \{Enc(RHS.pk_1, c_0, rnd_{0,0}), \ldots, Enc(RHS.pk_1, c_k, rnd_{0,k})\}$, $B = Enc(RHS.pk_2, C, RND_0)$ and a Naor-Yung proof π is provided with a disjunctive proof. The difference though that the D-shuffle also encrypts $A_{LHS} = \{Enc(LHS.pk, c_0, r_{0,0}), \ \ldots, \ Enc(LHS.pk, c_k, r_{0,k})\}$ for $r_{i,j}$ is also random,and provides PET for each pair of elements in A_{LHS} and its correspondence in A_{RHS}. All the zero knowledge proofs are submitted to the bulletin board.
- The D-shuffle shuffles the RHS as done in §4.3 by each mix-server j, re-encrypting the inputted ciphertexts it received, permuting the new ciphertexts and providing a 1-out-of-L re-encryption proof for elements of A_{RHS}, and the plaintext equivalence test for the homomorphic summation of the RHS values and the B value. The only addition we add is that the mix-server should also submit a proof that the RHS ciphertexts correspond to LHS ciphertexts since the randomization factors ratio for each ciphertext pair is known (i.e. $r_{i,j}$ and $rnd_{i,j}$) for each mix server j, i.e using equality between discrete logs proofs (See Appendix A). All proofs are submitted to the bulletin board.

The LHS is decrypted by a printing authority, which has the keys for the secret keys $LHS.sk$. That authority can be a threshold of authorities if the ElGamal encryption used is threshold based (maybe a threshold number of printers). The RHS is decrypted by the tallying authority and that can also be a threshold of authorities. Figure 3 describes the ballots.
Auditing of the LHS onion can be done by any one checking the plaintext equivalence tests on the bulletin board. For usability purposes, the voter can choose which authority they trust most for auditing the ballot. Auditing the whole ballot requires, revealing the decryption of both left and right hand side, as done in the original Prêt à Voter .

5 PAV is Receipt Free

Before we can start the Receipt Freeness proof we need to explain the concept of splitting ciphertexts. This concept will be used to simulate the challenge.

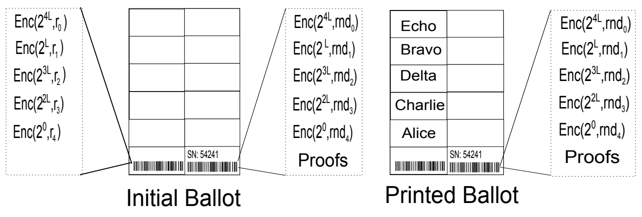

Fig. 3. Prêt à Voter : The Multiple Authority Ballot

Definition 3. *An encryption scheme is splittable if one can define an algorithm Split over its ciphertext as shown in Figure 5.*

Splitting can be done without knowing the randomization factor r or secret key sk. Both Exponential ElGamal and Paillier are splittable encryptions. Figure 5 shows Exponential ElGamal's Split Algorithm and in the Full Version of the paper we show Paillier to be Splittable.

$\boldsymbol{Split(CT, RND, \bar{M}, M)}$ **:** Takes as input a ciphertext $CT = Enc(M, r, pk)$, a randomization factor RND and another message $\bar{M}$. It outputs two ciphertexts CT_1 and CT_2 such that

- $CT_1 = Enc(M - \bar{M}, \frac{r}{RND}, pk)$
- $CT_2 = Enc(\bar{M}, \frac{r(RND-1)}{RND}, pk)$
- $CT = CT_1.CT_2$

Fig. 4. The Split Algorithm

The following explains the splitting procedure of Exponential ElGamal where $CT = (u, v) = (g^r, y^r g^M)$:

- Let $t_1 = \frac{1}{RND}$ and $t_2 = \frac{RND-1}{RND}$
- Let $m_1 = M - \bar{M}$ and $m_2 = \bar{M}$
- $u_i = u^{t_i} = g^{rt_i}$, $v_i = v^{t_i}(g^{-Mt_i+m_i}) = y^{rt_i} g^{m_i}$, for $i = \{1, 2\}$
- Let $CT_1 = (u_1, v_1)$, and $CT_2 = (u_2, v_2)$

Fig. 5. Exponential ElGamal is Splittable

In the proof we assume ballots are well formed. This allows us to drop all zero knowledge proofs other than Naor-Yung. One argues that this game model is for proving receipt freeness and these proofs of knowledge reveal minimum

$\mathcal{A}$		$\mathcal{S}$		$\mathcal{C}$
	$\xleftarrow{NY.PK}$	Initialize BB and DB	$\xleftarrow{NY.PK}$	Setup $NY.E$

Fig. 6. Setup Phase

information. Another argument, is that well formness of a ballot is captured in auditing and is an integrity related issue rather than privacy related.
The ballot referred to in this proof is one which has the ciphertexts and a proof of Naor-Yung as an onion. We explain the proof for the case of a single authority producing the ballot. We end the section with a discussion about how the proof can be extended for the multiple authority case.

Theorem 3. *If there exists an Adversary $\mathcal{A}$ that can break the receipt freeness of PAV then there exists a simulator $\mathcal{S}$ that breaks the IND-CCA2 security of Naor-Yung encryption.*

Proof. Assume we have a simulator $\mathcal{S}$ interacting with a challenger $\mathcal{C}$ with the intension of breaking the IND-CCA2 of Naor-Yung encryption. Assume we have an adversary $\mathcal{A}$ who breaks the RF-game model. The simulator $\mathcal{S}$ interacts with $\mathcal{C}$ and $\mathcal{A}$ as follows:

– Setup: $\mathcal{C}$ sets up $NY.E$ by running $NY.KeyGen(k)$. He gives $\mathcal{S}$ the public parameters $NY.pk = (pk_1, pk_2)$. The $\mathcal{S}$ initializes the bulletin board BB which is accessible to $\mathcal{A}$ as read only. He also creates a database of ballots DB that will contain (SN, $onion$, $candidates.order$, $vote$). Initially that database is empty. He passes to $\mathcal{A}$ the values in $NY.pk$. See Figure 6.
– RF.Phase[I]: $\mathcal{A}$ queries oracles REB,RHB,RUB,RCO,CNB,RBB,PRBB and Vote. We explain how the oracles are simulated (See Figure 7)
  • REB, RHB, and RUB: $\mathcal{S}$ receives a serial number SN of a ballot. He creates a new ballot by creating the proper encryptions given he has the public parameters required (i.e. $NY.PK$). He adds the onion in the database of (SN, $onion$, $candidates.order$, $vote$), with $vote$ being $null$. He sends either half or full ballot back to $\mathcal{A}$. If RUB is queried, he sends a used ballot ticked in the index requested. He updates the bulletin board to include it as a vote and has the candidate name registered in the database of ballots DB, as $vote = c.name$.
  • Vote: $\mathcal{S}$ receives the query as onion, and index. Checks the ballot on the bulletin board if it has been used before, checks the $vote$ element in the database, if the ballot exists but the vote does not, updates bulletin board, and updates database to have $vote = c.name$. If the ballot does not exist in database then it most be created by the adversary, so query the decryption oracle from $\mathcal{C}$ to get the information of the ballot. Update the database and the bulletin board as required. At the end of this oracle the adversary should get one of two responses: either the "ballot has been used" or "the vote accepted, and a receipt". If it is the latter response then the adversary should be able to see an updated bulletin board.

$\mathcal{A}$		$\mathcal{S}$		$\mathcal{C}$
Oracle REB,RHB, RUB:				
	$\xrightarrow{REB:SN}$	Create $Ballot$;		
	$\xrightarrow{RHB:SN}$	If RUB; vote = $c.name$, update BB, and DB		
	$\xrightarrow{RUB:SN}$	Else; Update DB with vote=null;		
		If REB; response = empty ballot;		
		If RHB; response = half ballot;		
		If RUB; response = used ballot;		
	$\xleftarrow{response}$			
Oracle Vote:				
	$\xrightarrow{Vote:Onion,index}$	If $Ballot \in DB$, vote$\neq$ $null$, $Ballot \in BB$;		
	$\xleftarrow{response}$	response = Ballot used.		
		If $Ballot \in DB$, vote= $null$, $Ballot \notin BB$;		
		Update BB, Update DB vote= $c.name$		
		If $Ballot \notin DB$	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB, Update BB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
	$\xleftarrow{response}$	response = receipt,vote accepted		
Oracle RCO:				
	$\xrightarrow{RCO:RHS}$	If $Ballot \in DB$;		
	$\xleftarrow{response}$	response =candidate.order		
		If $Ballot \notin DB$	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
	$\xleftarrow{response}$	response =candidate.order		
Oracle CNB:				
	$\xrightarrow{CNB:Ballot}$	If $Ballot \in DB$		
	$\xleftarrow{response}$	response = ballot exist		
		If $Ballot \notin DB$	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
	$\xleftarrow{response}$	response = ballot added		
Oracle RBB:				
	$\xrightarrow{RBB}$	Result = Tally BB		
	$\xleftarrow{Results}$			
Oracle PRBB:				
	$\xrightarrow{PRBB:\{Ballots\}}$	If $allBallot \in DB$;		
	$\xleftarrow{Results}$	Result=Tally BB		
		If $allBallot \notin DB$;		
		Do :	$\xrightarrow{Dec:Ballot}$	candidates.order =
		Update DB	$\xleftarrow{candidate.order}$	NY.Dec(ballot,..)
		Until: all ballots in DB		
	$\xleftarrow{Results}$	Result=Tally BB		

Fig. 7. Phase I and Phase II

- The oracle RCO is handled in a similar way, if the ballot is in the database return the candidate order otherwise query the decryption oracle, update the database then return the candidate order.
- For the RBB, the simulator tallies the bulletin board and returns the result.
- For the PRBB oracle, the simulator $\mathcal{S}$ checks the database to reveal information, if any ballot does not exist in the database it retrieves the information using the decryption oracle and adds it to DB.

– RF.Challenge: $\mathcal{A}$ picks two serial numbers i and j, where the two ballots have not been queried before in REB, RHB, RUB, Vote and RCO and sends

$\mathcal{A}$	$\mathcal{S}$		$\mathcal{C}$
$\xrightarrow{i,j}$	$M_1 = 2^L + 2^{2L}$		
	$M_2 = random$	$\xrightarrow{M_1,M_2}$	$NY.CT_b = NY.Enc(M_b, NY.pk)$
	Create Ballot i with	$\xleftarrow{C_b}$	$NY.CT_b = (c_1, c_2, \pi)$
	$(E_{Alice}, E_{Bob}) = Split(c_1, 2^L, RND, M_1)$		
	$Onion = (E_{Alice}, E_{Bob}, \pi, c_2)$		
	Create Ballot j as in RUB		
	Toss fair coin		
	If coin = head;		
	i votes for Alice		
	j votes for Bob		
	If coin = tail;		
	i votes for Bob		
	j votes for Alice		
$\xleftarrow{Receipt_i, Receipt_j}$	Update BB and DB		

Fig. 8. Challenge Phase

them to $\mathcal{S}$. Assume the code of Alice was 2^L and of Bob 2^{2L}. $\mathcal{S}$ chooses two messages $M_1 = 2^L + 2^{2L}$ and M_2 is random. It sends the messages to $\mathcal{C}$ who responds with $NY.CT_b = NY.Enc(M_b, NY.pk) = (c_1, c_2, \pi)$ where $b \in \{1, 2\}$. $\mathcal{S}$ computes E_{Alice}, and E_{Bob} using the split algorithm. Assuming $b = 1$, the elements E_{Alice}, E_{Bob}, and $NY.CT_b$ will form a valid onion because $E_{Alice} = Enc(2^L, \frac{r}{RND}, pk_1)$ is a valid encryption of Alice's code and $E_{Bob} = Enc(2^{2L}, \frac{r(RND-1)}{RND}, pk_1)$ is a valid encryption of Bob's code. The last part of the onion is the Naor-Yung transformation $NY.CT_b$ that guarantees CCA2 security notion for the whole ballot. $\mathcal{S}$ creates a first ballot i with order {Alice, Bob} and the onion order is E_{Alice}, E_{Bob}. He then creates a second ballot j normally using the public keys he knows, he tosses a fair coin if the toss is equal to head, he uses i to be a vote for Alice, and j to be a vote for Bob, otherwise he swaps the votes. Finally, he updates the database and bulletin board accordingly. He sends the receipts to $\mathcal{A}$. See Figure 8.

– RF.Phase[II]: This phase is similar to RF.Phase [I] with the condition that the challenge is not queried in REB, RHB, RUB, Vote and RCO.
– RF.Guess: Assuming that $b = 1$ then the ballots are well formed and $\mathcal{A}$ returns the right serial number for the vote Alice, otherwise the ballots are not well formed and the adversary aborts. $\mathcal{S}$ receives the serial number, if it is what he expected then he returns to the $\mathcal{C}$ that $b = 1$ otherwise $b = 2$.

Assuming the adversary $\mathcal{A}$ wins the RF-game model with non-negligible advantage, the $\mathcal{S}$ wins the IND-CCA2 game model, since there is no way that the $\mathcal{A}$ will respond i without the ballots being well formed implying $b = 1$ and otherwise the adversary $\mathcal{A}$ will abort, implying $b = 2$

Multiple Authority Extension: The game model can be extended to simulate ballots created by multiple authorities. The idea is to drop the 1-out-of-L Re-encryption proofs for the same reason behind dropping the disjunctive proofs. The simulator, in the RF.Setup stage, runs an encryption key generation algorithm to create a pair of keys for the LHS only. He gives the public keys he received from $\mathcal{C}$ to $\mathcal{A}$ together with the public key of LHS. The oracle Reveal Candidate Order (RCO) can be used to query either LHS or RHS. However, given the private key of the LHS is known to the simulator he will not need to query the decryption oracle. The rest of the oracles and the challenge are simulated as in the single authority case.

6 Conclusion

Prêt à Voter has been introduced in [Rya05]. The general idea behind the scheme is to provide an end-to-end verifiable, receipt free, paper based, e-voting scheme. Several formal method analysis exist in the literature to prove the properties of the scheme [XSHT08,RBH$^+$09]. In this paper we provide a proof of receipt freeness of Prêt à Voter using provable security game models. We defined a new game model that presents receipt freeness. We discussed the Prêt à Voter latest version introduced in [XCH$^+$10], and showed how their construction together with a special verifiable shuffle, the D-shuffle, construct a receipt free Prêt à Voter scheme. The D-shuffle proposed together with ElGamal encryption form a Naor and Yung transformation, causing the ballot to be encrypted in an IND-CCA2 secure manner. We proved that the existence of an adversary that wins the receipt freeness game model implies the existence of a simulator that can break the IND-CCA2 security of Naor-Yung transformation. Finally, the "D-shuffle" helps in providing the option of having multiple authorities for creating the ballot.

Acknowledgment

The authors would like to thank Wafa Alansari, Marc Pouly and Sasa Radomirovic for there valuable comments. Special thanks for Sriram Srinivasan for his extensive review of the paper and for providing important feedback. Furthermore, this research is funded by the FNR (National Research Fund) Luxembourg under project SeRVTSC09/IS/06

References

[Abe99] M Abe. Mix-networks on permutation networks. In *Asiacrypt'99*, volume 1716 of *LNCS*, pages 258–273. Springer-Verlag, 1999.

[AMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *EVT/WOTE'09*. USENIX Association, 2009.

[BCP$^+$11] David Bernhard, Vèronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *(ESORICS'11)*, 2011.

[BFP$^+$01] O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. In *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing*, pages 274–283. ACM press, 2001.

[BHM08] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08*, pages 195–209. IEEE Computer Society, 2008.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.

[CEG88] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141. Springer, 1988.

[CEGP87] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. In *CRYPTO'86*, volume 263 of *LNCS*, pages 200–212. Springer, 1987.

[CP93] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[Dag07] Dagstuhl Accord. Participants of the Dagstuhl Conference on Frontiers of E-Voting. http://www.dagstuhlaccord.org/, 2007.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, jul 2009.

[FCS06] Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *IAVoSS Workshop On Trustworthy Elections*, 2006.

[FMOS02] J. Furukawa, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting protocol based on shuffling. In *Financial Cryptography*, LNCS. Springer-Verlag, 2002.

[FS02] J. Furukawa and K. Sako. An efficient protocol for proving a shuffle. In *Crypto'01*, volume 2139 of *LNCS*, pages 368–387. Springer-Verlag, 2002.

[GL07] Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In *In proceedings of Practice and Theory in Public Key Cryptography - PKC 07, LNCS 4450*, pages 377–392, 2007.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 539–556, Berlin, Heidelberg, 2000. Springer-Verlag.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05*, pages 61–70. ACM Press, 2005. See also http://www.rsa.com/rsalabs/node.asp?id=2860.

[KRS10] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[MA99] Jakobsson Markus and Juels Ari. Millimix: Mixing in small batches. Technical report, Center for Discrete Mathematics, 1999.

[MN06] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy . In *Crypto'06*, volume 4117 of *LNCS*, pages 373–392. Springer-Verlag, 2006.

[MN10] Tal Moran and Moni Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.*, 13:16:1–16:43, March 2010.

[Nea87] Koblitz Neal. *A course in number theory and cryptography.* Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[Nef01] Andrew Neff. A verifiable secret shuffle and its application to evoting. In *CCS'01*, pages 116–125. ACM press, 2001.

[NSNK05] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. A provably secure and efficient verifiable shuffle based on a variant of the paillier cryptosystem. In *Journal of Universal Computer Science*, page 9861010. ACM press, 2005.

[NY95] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *In Proc. of the 22nd STOC*, pages 427–437. ACM Press, 1995.

[OT04] Takao Onodera and Keisuke Tanaka. A verifiable secret shuffle of pailliers encryption scheme. Technical report, Tokyo Institute of Technology, 2004.

[Ped91] Torben P. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *EUROCRYPT'91*, number 547 in LNCS, pages 522–526. Springer, 1991.

[RBH$^+$09] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: a voter-verifiable voting system. *Trans. Info. For. Sec.*, 4:662–673, December 2009.

[RS06] Peter Y. A. Ryan and Steve A. Schneider. Prêt à voter with re-encryption mixes. In *ESORICS*, pages 313–326, 2006.

[RT09] Peter Y. A. Ryan and Vanessa Teague. Pretty Good Democracy. In *Proc. of the 17th Security Protocols Workshop*, LNCS. Springer, 2009.

[Rya05] Peter Y. A. Ryan. A variant of the chaum voter-verifiable scheme. In *Proceedings of the 2005 workshop on Issues in the theory of security*, WITS '05, pages 81–88, New York, NY, USA, 2005. ACM.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:543, 1999.

[Sch89] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.

[SK95] K. Sako and J. Kilian. A practical solution to the implementation of voting booth. In *Eurocrypt'95*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

[Sma04] N. Smart. *Cryptography: An Introduction.* Mcgraw-Hill College, 2004.

[Wik05a] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *ASIACRYPT 05*, volume 3788 of *LNCS*, page 273292. Springer-Verlag, 2005.

[Wik05b] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. Technical report, IACR, 2005.

[Wik06] Douglas Wikström. Simplified submission of inputs to protocols. Cryptology ePrint Archive, Report 2006/259, 2006. http://eprint.iacr.org/.

[XCH$^+$10] Zhe Xia, Chris Culnane, James Heather, Hugo Jonker, Peter Y. A. Ryan, Steve Schneider, and Sriramkrishnan Srinivasan. Versatile Prêt à Voter : Handling Multiple Election Methods with a Unified Interface. In *INDOCRYPT*, volume 6498 of *LNCS*, pages 98–114. Springer-Verlag, 2010.

[XSH$^+$07] Z. Xia, S. Schneider, J. Heather, P. Ryan, D.Lundin, R. Peel, and P. Howard. Prêt à Voter: All-In-One. In *IAVoSS (WOTE'07)*, 2007.

[XSHT08] Zhe Xia, Steve A. Schneider, James Heather, and Jacques Traoré. Analysis, improvement and simplification of prêt à voter; voter with paillier encryption. In *Proceedings of the conference on Electronic voting technology*, pages 13:1–13:15, Berkeley, CA, USA, 2008. USENIX Association.

A Non-interactive Zero knowledge proofs

Knowledge of discrete logs: Proving knowledge of x, given h where $h \equiv g^x \bmod p$ [CEGP87,CEG88,Sch89].

Sign. Given x, select a random nonce $w \in_R \mathbb{Z}_q^*$ and compute, Witness $g' = g^w \bmod p$, Challenge $c = \mathcal{H}(g') \bmod q$ and Response $s = w + c \cdot x \bmod q$. Output Signature (g', s)

Verify. Given h and signature (g', s), check $g^s \equiv g' \cdot h^c \pmod{p}$, where $c = \mathcal{H}(g') \bmod q$.

A valid proof asserts knowledge of x such that $x = \log_g h$; that is, $h \equiv g^x \bmod p$.
Equality between discrete logs: Proving knowledge of the discrete logarithm x to bases $f, g \in \mathbb{Z}_p^*$, given h, k where $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$ [Ped91,CP93].

Sign. Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute Witnesses $f' = f^w \bmod p$ and $g' = g^w \bmod p$, Challenge $c = \mathcal{H}(f', g') \bmod q$ and Response $s = w + c \cdot x \bmod q$. Output signature as (f', g', s)

Verify. Given f, g, h, k and signature (f', g', s, c), check $f^s \equiv f' \cdot h^c \pmod{p}$ and $g^s \equiv g' \cdot k^c \pmod{p}$, where $c = \mathcal{H}(f', g') \bmod q$.

A valid proof asserts $\log_f h = \log_g k$; that is, there exists x, such that $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$. Note that this proof of knowledge can be used to prove equality of plaintexts for two ElGamal ciphertext encryptions such a proof is referred to as plaintext equivalence test (PET).
Disjunctive proof of equality between discrete logs Let $h = g^y$. Given $(a, b) = (g^x, h^x \cdot g^m)$ contains message m, prove that $m \in \{\mathsf{min}, \dots, \mathsf{max}\}$ for some parameters $\mathsf{min}, \mathsf{max} \in \mathbb{N}$ [CDS94].

Sign. Given (a, b) such that $a \equiv g^x \bmod p$ and $b \equiv h^x \cdot g^m \bmod p$ for some nonce $x \in \mathbb{Z}_q^*$, where plaintext $m \in \{\mathsf{min}, \dots, \mathsf{max}\}$. For all $i \in \{\mathsf{min}, \dots, m-1, m+1, \dots, \mathsf{max}\}$, compute challenge $c_i \in_R \mathbb{Z}_q^*$, response $s_i \in_R \mathbb{Z}_q^*$, and witnesses $a_i = g^{s_i}/a^{c_i} \bmod p$ and $b_i = h^{s_i}/(b/g^i)^{c_i} \bmod p$. Select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $a_m = g^w \bmod p$ and $b_m = h^w \bmod p$, challenge $c_m = \mathcal{H}(a, b, a_{\mathsf{min}}, b_{\mathsf{min}}, \dots, a_{\mathsf{max}}, b_{\mathsf{max}}) - \sum_{i \in \{\mathsf{min}, \dots, m-1, m+1, \dots, \mathsf{max}\}} c_i \pmod{q}$ and response $s_m = w + x \cdot c_m \bmod q$. To summarise, we have the Witnesses $(a_{\mathsf{min}}, b_{\mathsf{min}}), \dots, (a_{\mathsf{max}}, b_{\mathsf{max}})$, Challenge $c_{\mathsf{min}}, \dots, c_{\mathsf{max}}$ and Response $s_{\mathsf{min}}, \dots, s_{\mathsf{max}}$. Output signature of knowledge (a_i, b_i, c_i, s_i) for all $i \in \{\mathsf{min}, \dots, \mathsf{max}\}$.

Verify. Given (a, b) and $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \dots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$, for each $\mathsf{min} \le i \le \mathsf{max}$ check $g^{s_i} \equiv a_i \cdot a^{c_i} \pmod{p}$ and $h^{s_i} \equiv b_i \cdot (b/g^i)^{c_i} \pmod{p}$. Finally, check $\mathcal{H}(a, b, a_{\mathsf{min}}, b_{\mathsf{min}}, \dots, a_{\mathsf{max}}, b_{\mathsf{max}}) \equiv \sum_{\mathsf{min} \le i \le \mathsf{max}} c_i \pmod{q}$.

A valid proof asserts that (a, b) contains the message m where $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$.
1-out-of-L ElGamal Re-encryption Proof: Let $h = g^y$. Given $(u_i, v_i) = (g^x g^\zeta, h^x \cdot h^\zeta \cdot g^m)$ is a re-encryption of $(u, v) = (g^x, h^x \cdot g^m)$ for a random $\zeta \in \mathbb{Z}_p^*$. Prove that (u_i, v_i) belongs to the list $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ [HS00].

Sign. Select random values $d_1, \ldots, d_n, r_1, \ldots, r_n \in \mathbb{Z}_p^*$. Compute $a_t = (\frac{u_t}{u})^{d_t} g^{r_t}$, $b_t = (\frac{v_t}{v})^{d_t} h^{r_t}$ where $t \in \{1, \ldots, i-1, i+1, \ldots, n\}$. Choose randomly a nounce $\omega \in \mathbb{Z}_p^*$. Let $a_i = g^\omega$ and $b_i = h^\omega$. Compute challenge $c = \mathcal{H}(E||a_1||\ldots||a_n||b_1||\ldots||b_n)$ where $E = (u||v||u_1||v_1||\ldots||u_n||v_n)$. Compute $d_i = c - \sum_{t=1, t\neq i}^{n} d_y$. Compute $r_i = \omega - \zeta d_i$ then Witnesses $d_1, \ldots, d_n$, Challenge c and Response $r_1, \ldots, r_n$. Output signature of knowledge (r_t, d_t) where $t \in [1, n]$

Verify. Check $\sum_{t=1}^{n} d_t = \mathcal{H}(E||(\frac{u_1}{u})^{d_1} g^{r_1}||\ldots||(\frac{u_n}{u})^{d_n} g^{r_n}||(\frac{v_1}{v})^{d_1} g^{r_1}||\ldots||(\frac{v_n}{v})^{d_n} g^{r_n})$

A valid proof asserts that $(u_i, v_i) \in \{(u_1, v_1), \ldots, (u_n, v_n)\}$.

B Encryption Schemes Security Notions [Sma04]

We explain IND-CPA, IND-CCA, and IND-CCA2 in this section. In an IND-CPA $\mathcal{A}$ and $\mathcal{C}$ agree on the ENC scheme, the plaintext message space M and ciphertext message space C they want to challenge. We say that ENC is IND-CPA secure if and only if there exist no polynomial time adversary that can win the IND-CPA game. Assume the encryption algorithm is E. The IND-CPA game is described as follows:

- $\mathcal{C}$ sets up the system by creating a public key pk, and a secret key sk . $\mathcal{C}$ gives pk to $\mathcal{A}$ and retains sk.
- $\mathcal{A}$ chooses two messages M_1 and M_2. They should be the same size or the shorter message should be padded to equalize the size. $\mathcal{A}$ sends the two messages to $\mathcal{C}$.
- $\mathcal{C}$ randomly chooses $b \in \{1, 2\}$ and encrypts a message $C_b = E(M_b, pk)$. In other words one of the messages is chosen randomly and encrypted. C_b is sent to $\mathcal{A}$ as the challenge.
- *Adam* uses the ciphertext C_b and all his computational ability to choose a $\bar{b} \in \{0, 1\}$ that matches b. If $\bar{b} = b$ then $\mathcal{A}$ wins the game else $\mathcal{A}$ loses.

The advantage of winning the game is defined by how much better $\mathcal{A}$ can do than a simple random guess (which has a probability 1/2 of being right). In other words, $Adv_{CPA}(k) = |Pr[b = \bar{b}] - 1/2|$ where k is the security parameter used in setting up the encryption scheme.

Definition 4. (Indistinguishable Chosen Plaintext Attack) *An encryption scheme is IND-CPA secure if and only if $Adv_{CPA}(k) < \varepsilon$ where ε is negligible and k is the security parameter used in setting up the system.*

In the literature there are more powerful security notions for encryption schemes other than IND-CPA. For example, we can give the adversary access to a decryption oracle that he queries a number of times before and after the challenge, as long as he does not use the challenge ciphertext for issuing a query. If $\mathcal{A}$ is given access to such an oracle before the challenge only then the encryption scheme is said to be IND-CCA secure, (i.e. Non-adaptive), otherwise the scheme is IND-CCA2 secure, (i.e. Adaptive).