USENIX Journal of Election Technology and Systems (JETS)
Volume 1, Number 1 • August 2013

Proving Prêt à Voter Receipt Free using Computational Security Models*

Dalia Khader, Peter Y. A. Ryan, Qiang Tang
Interdisciplinary Centre for Security Reliability and Trust, SnT
and Université du Luxembourg

Abstract

Prêt à Voter is a supervised, end-to-end verifiable voting scheme. Informal analyses indicate that, subject to certain assumptions, Prêt à Voter is receipt free, i.e. a voter has no way to construct a proof to a coercer of how she voted. In this paper we propose a variant of Prêt à Voter and prove receipt freeness of this scheme using computational methods. Our proof shows that if there exists an adversary that breaks receipt freeness of the scheme then there exists an adversary that breaks the IND-CCA2 security of the Naor-Yung encryption scheme.

We propose a security model that defines receipt freeness based on the indistinguishability of receipts. We show that in order to simulate the game we require an IND-CCA2 encryption scheme to create the ballots and receipts. We show that, within our model, a non-malleable onion is sufficient to guarantee receipt freeness. Most of the existing Prêt à Voter schemes do not employ IND-CCA2 encryption in the construction of the ballots, but they avoid such attacks by various additional mechanisms such as pre-commitment of ballot material to the bulletin board, digitally signed ballots etc. Our use of the Naor-Yung transformation provides the IND-CCA2 security required.

1 Introduction

The role of voting is to ascertain the collective intent of the electorate. Voting systems are required to guarantee that the announced result is a true reflection of the intent of the electorate, given appropriate rules for the expression of preferences and computing the outcome.

In order to ensure that voters can express their intent freely, voting systems are usually required to guarantee *ballot privacy*, i.e. that no-one other than the voter herself will know how she cast her vote. Failure to guarantee this will open voters up to being deflected from expressing their true intent via either vote buying or coercion attacks. Note that it is not enough for the voting system to provide this property, it is essential that it is perceived by (vast majority of) the electorate to do so.

Security experts and cryptographers took an interest in the challenge of minimizing the level of trust in officials or in the technology used in the process. This gave rise to the notion of *voter-verifiability* or *full auditiability*, i.e. to provide every voter with the means to confirm that her vote is accurately included in the tabulation, while at

*Funded by the FNR (National Research Fund) Luxembourg under project SeRVTSC09/IS/06

the same time avoiding any violation of ballot secrecy. Typically voter-verifiability is achieved by providing each voter with an encrypted version of her vote. She can later use this to confirm that her (encrypted) vote is entered into the tabulation process. Various implementations of this idea have been proposed, [12, 25, 31], and in section 3 we will outline a particular approach: Prêt à Voter .

These two requirements, one a form of verifiability and the other of a form of secrecy, are clearly in conflict and achieving them simultaneously with minimal trust assumptions in a hostile environment has proved immensely challenging. In particular, the mechanisms that provide voter-verifiability (the encrypted ballots, the bulletin board etc.) can, if not carefully designed, introduce threats to ballot privacy.

This conflict has given rise to the introduction of more elaborate notions of ballot secrecy, namely *receipt-freeness* and *coercion resistance*. We will go into more precise definitions of these notions later in the paper, but these new properties can be thought of informally as guaranteeing ballot secrecy against increasingly powerful attackers. Ballot secrecy ensures that a passive attacker, one who simply observes the unfolding of the voting protocol, cannot determine how any individual voter cast her vote. Receipt-freeness (first defined by Benaloh and Tuinstra [11]) takes account of the fact that a voter might cooperate with an attacker to construct a proof of how she voted, using the data generated in the execution of the voting protocol. In particular the voter may be able to reveal certain data that ordinarily is assumed secret, for example passwords, credentials or randomization factors using in encryption. For coercion resistance we assume further that the attacker may interact with the voter at various stages of the voting protocol: before, during and after.

Prêt à Voter is an end–to–end verifiable scheme that gives the voters a receipt when they submit their vote to verify that their votes were never modified. Informal analysis of Prêt à Voter indicates that, subject to various assumptions, it is receipt free. however, certain subtle attacks on Prêt à Voter original version [27], made it not coercion resistance (§4.1). Variants of Prêt à Voter were proposed after to address these attacks [26]. In this paper we focus on analyzing the receipt freeness property using computational models. Strengthening our models and proofs to include different levels of coercion resistance is kept for future studies.

1.1 Contribution

Our work is the first game based analysis of the receipt freeness of (a variant of) Prêt à Voter based on indistinguishablity of receipts. Formal methods (symbolic) techniques have been used earlier on to analyze the same notion in Prêt à Voter [24, 33], but these methods do not capture non-malleability of the ballot.

It should be noted that the standard versions of Prêt à Voter assume that blank receipts are posted to the Bulletin Board and signed ahead of the election. Hence, casting vote corresponds in effect to filling in the vote information in an existing blank ballot. The approach counters related plaintext style attacks on the tabulation that involve the attacker posting receipts that are constructed in some way from real receipts. Our model does not assume such a mechanism, hence the need to ensure non-malleability of receipts. In this model we use the Noar-Yung construction as this meshes nicely with some existing constructions to enable print-on-demand of Prêt à Voter ballots. These constructions involved encrypting the candidate order under two, distinct public keys,

one for the tabulation tellers and one for the printer, along with Zero-Knowledge proofs of the correspondence of the orders.

We will see that the model presented in this paper captures many of the notions mentioned earlier, essentially those that can be classed as receipt-freeness, but fails to capture certain attacks that fall into the category of coercion-resistance, e.g randomisation attacks. Capturing notions such as receipt-freeness and coercion resistance has proved very challenging in any formalism, but poses particular challenges in the context of computational models. We note also that modelling Prêt à Voter poses new challenges due to the special way that encrypted ballots are created: in contrast to most voter-verifiable schemes, Prêt à Voter does not directly encrypt the voter's choice but rather pre-prepares an encryption of a randomized frame of reference in which the vote is encoded.

Finally, we should mention that we provide an abstract description of Prêt à Voter in §3. A voting scheme that fits that description can be proven secure in our security model in §5. If we make changes on the system level a new model needs to be provided to analyze the system, however if we make changes on the construction level only, a new proof of security needs to be provided but the same security model can be used[1].

1.2 Related Work

Although e-voting schemes use cryptography to ensure various levels of privacy, little work has been done on proving them secure using computational models. The complexity of e–voting schemes and the different security requirements needed, makes it extremely challenging to find one computational model suitable for all voting schemes, therefore researchers proposed several models each suitable for specific voting scheme, we list some below.

Okamoto provided the first provable secure receipt free voting scheme [20]. Moran and Naor [17, 18] aimed for the everlasting–privacy property in their voting schemes. They use Universal Composability framework in their proofs and their constructions are based on commitment schemes. Küsters et al. analyzes four e-voting schemes using simulation based models: Bingo, TheeBallot, VAV, and Scantegrity II [14–16]. The main idea in all papers is that a coercer can distinguish whether a vote has followed his orders or have voted as to his intended goal and that is done by running a counter-strategy. Bernhard et al. [2] proposed a voting-friendly encryption scheme that can be used in the Helios voting system and proved that such a scheme is needed to prove ballot secrecy using game based models. Helios is a remote voting scheme and unlike Prêt à Voter it does not require going to a voting booth.

Work has been done on proving receipt freeness in voting schemes using symbolic analysis [1, 9, 13], §2.1 explains the difference between the two models briefly.

2 Preliminary Concepts

2.1 Provable Security

In formal method analysis the cryptographic primitives are represented by function symbols considered as black–boxes, and the analysis assumes perfect cryptography

[1] Preliminary version of this paper is published on ePrint Archive: Report 2011/594.

(e.g. the algorithms "Encrypt" and "Decrypt" are treated as symbolic operations and that means agents can decrypt only if they know the right keys, but cannot otherwise "crack" encrypted messages). In computational models the cryptographic primitives are functions from bit–strings to bit–strings, and the adversary is any probabilistic Turing machine. Even though the computational models seem more realistic, but they complicate the proofs, and it is hard to study a large system using them.

In this paper we study receipt freeness of Prêt à Voter using a computational model referred to as *provable security* paradigm. Evaluating the security of a protocol in the *provable security* paradigm proceeds in two steps.

1. The first step is to design a security model for a category of cryptographic protocols. The security model usually takes the form of an adversary vs challenger game, where the adversary represents the potential attacker which may break the protocol in practice and the challenger represents the honest parties in the protocol execution. In the game, the adversary interacts with the challenger through some oracles, which reflects the privileges that he can have in practice. Moreover, an advantage will be defined to indicate the condition that the adversary has broken the security of the protocol and won the game. Section 4 provides a security model for Prêt à Voter receipt freeness.
2. Given a protocol and the corresponding security model, the proof is essentially to show that if any adversary can win the game with a non-negligible advantage then it can actually be used as a subroutine to solve a hard problem, e.g. discrete logarithm problem. If we can make the assumption that the hard problem is intractable then we can draw the conclusion that the protocol is secure in the security model based on the assumption (See §5).

As an example, security models have been defined for public-key encryption schemes in the next subsection and the same example will be used in our security proof of receipt freeness (See §5). With respect to provable security, one important notion is negligibility, formally defined as follows.

Definition 2.1 *Function $P(k) : \mathbb{Z} \to \mathbb{R}$ is said to be negligible with respect to k if, for every polynomial $f(k)$, there exists an integer N_f such that $P(k) < \frac{1}{f(k)}$ for all $k \geq N_f$.*

2.2 Security Notions for Public-key Encryption

A public-key encryption scheme consists of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, defined as follows. $\mathsf{KeyGen}(\lambda)$ takes a security parameter λ as input and outputs a public/private key pair (pk, sk). $\mathsf{Enc}(M, pk)$ takes a message M and the public key pk as input, and outputs a ciphertext C. $\mathsf{Dec}(C, sk)$ takes a ciphertext C and the private key sk as input, and outputs a plaintext M or an error symbol $\perp$.

For public-key encryption schemes, the strongest security notion is IND-CCA2 (indistinguishability under adaptive chosen ciphertext attack). Informally, this property guarantees that an (adaptive and active) attacker will not be able to learn any information about the plaintext in a ciphertext c, even if it can decrypt any ciphertext except for c. This property is essential when applying public-key encryption schemes to build interactive security protocols, where an attacker may try to learn private information through interactions with honest parties.

Definition 2.2 *A public-key encryption scheme achieves IND-CCA2 security if any*

polynomial time attacker only has negligible advantage in the attack game, shown in Figure 1. Note that the advantage is defined to be $|\Pr[b' = b] - \frac{1}{2}|$.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker is given pk and can issue a polynomial number of decryption queries with any input: Given C, the challenger returns $\mathsf{Dec}(C, sk)$. At some point, the attacker chooses M_0, M_1 of equal length and sends them to the challenger for a challenge.
3. *Challenge*. The challenger selects $b \in_R \{0, 1\}$ and returns $C_b = \mathsf{Enc}(M_b, pk)$ as the challenge.
4. *Phase 2*. The attacker can issue a polynomial number of decryption oracle queries with any input except for C_b.
5. *Guess*: At some point the attacker terminates Phase 2 by outputting a guess b' for b.

Figure 1: IND-CCA2 Game

In Definition2.2, if we remove *Phase 2* in the attack game then it becomes the definition for IND-CCA1 (indistinguishability under chosen ciphertext attack). Furthermore, if we completely disallow the attacker to access the decryption oracle then it becomes the standard IND-CPA (indistinguishability under chosen plaintext attack).

The above definition for IND-CCA2 is the standard one, but it has other equivalent forms. Below, we propose a different security model (i.e. IND-CCA2† security) for public key encryption schemes and show that this new security model is equivalent to the standard IND-CCA2 (Appendix A). The information format in this definition matches with the data format in our voting scheme, hence, this new model facilitates our security analysis in Section 5.

Definition 2.3 *A public-key encryption scheme achieves IND-CCA2† security if any polynomial time attacker only has negligible advantage in the attack game, shown in Figure 2. Note that the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker is given pk and can issue a polynomial number of decryption queries with any input: Given C, the challenger returns $\mathsf{Dec}(C, sk)$. At some point, the attacker chooses M_0, M_1 of equal length and sends them to the challenger for a challenge.
3. *Challenge*. The challenger selects $d \in_R \{0, 1\}$. If $d = 0$, the challenger returns $E_0 = (\mathsf{Enc}(M_0, pk), \mathsf{Enc}(M_1, pk))$ as the challenge, otherwise returns $E_1 = (\mathsf{Enc}(M_1, pk), \mathsf{Enc}(M_0, pk))$ as the challenge.
4. *Phase 2*. The attacker can issue a polynomial number of decryption queries with any input except for the two ciphertexts in E_d.
5. *Guess*: At some point the attacker terminates Phase 2 by outputting a guess d' for d.

Figure 2: IND-CCA2† Game

It is known that if a public-key encryption scheme is IND-CCA2 secure, then it must be IND-CPA secure. In appendix A, we prove the following theorem on the equivalence between IND-CCA2 and IND-CCA2†. This theorem allows us to employ an IND-CCA2 secure scheme and prove the security in an IND-CCA2† security model.

Theorem 2.4 *A public-key encryption scheme* (KeyGen, Enc, Dec) *is IND-CCA2 secure if and only if it is IND-CCA2† secure.*

2.3 Zero Knowledge Proofs

Loosely speaking, a zero knowledge proof is a cryptographic primitive that remarkably provides a convincing proof that an assertion holds without yielding any further information. In this paper we use non–interactive Zero Knowledge proofs (NIZK). Given language L_R defined by NP–relation R. The pair $(x, w) \in R$ if w is a witness that $x \in L_R$. A proof of system L_R is given by a pair of algorithms $(Prove, Verify)$. The prover and verifier both have an element $x \in L_R$ as input. In addition the prover has witness w such that $R(x, w) = 1$. The prover computes and sends a proof π to the verifier. The verifier outputs a decision to accept or reject the proof π. The requirements of the proofs is to be sound (if $x \notin L_R$ the verifier should reject π with high probability), correct (if $x \in L_R$ then verifier should accept proof) and zero knowledge (Nothing other than the validity of the assertion is revealed[2]).

2.4 Naor–Yung Transformation

The Naor-Yung IND-CCA2 construction [19] needs two ingredients: an IND-CPA encryption scheme (KeyGen, Enc, Dec) and a sound zero-knowledge proof system (Prove, Verify) that can prove that two ciphertexts encrypt the same message (i.e. A plaintext equivalence proof of different public keys, See Appendix B.3). In more detail, the Naor-Yung Transformation produces a scheme $NY.E = (NY.\mathsf{KeyGen}, NY.\mathsf{Enc}, NY.\mathsf{Dec})$ as follows.

- $NY.\mathsf{KeyGen}(\lambda)$: Take security parameter λ as input and run KeyGen twice to produce two key pairs $(sk_1, pk_1) = \mathsf{KeyGen}(\lambda)$ and $(sk_2, pk_2) = \mathsf{KeyGen}(\lambda)$. The secret key $NY.sk = (sk_1, sk_2)$ and public key $NY.pk = (pk_1, pk_2)$.
- $NY.\mathsf{Enc}(m, NY.pk)$: Compute $c_1 = \mathsf{Enc}(m, pk_1)$, $c_2 = \mathsf{Enc}(m, pk_2)$, and $\pi = \mathsf{Prove}(m, pk_1, pk_2, r_1, r_2, c_1, c_2)$, where r_1, r_2 represents the randomness used in constructing c_1 and c_2 respectively. The final ciphertext is (c_1, c_2, π).
- $NY.\mathsf{Dec}(c_1, c_2, \pi)$: If $\mathsf{Verify}(c_1, c_2, \pi) = 1$ then output $m = \mathsf{Dec}(c_1, sk_1)$; otherwise output an error symbol $\perp$.

Theorem 2.5 ***(Sahai [28]):*** *If the zero knowledge proof system* (Prove, Verify) *is a proof of knowledge and has uniquely applicable proofs (essentially each proof can only be used to prove one statement) and if the encryption scheme* (KeyGen, Enc, Dec) *is IND-CPA then applying Naor-Yung transformation gives an IND-CCA2 secure encryption scheme.*

In Appendix B we provide examples of the zero knowledge proofs that can be used with exponential ElGamal to obtain a Naor–Yung encryption.

[2] A proof is zero knowledge if there exist a simulator capable to produce transcripts indistinguishable from the real transcripts.

3 Prêt à Voter

The Prêt à Voter approach to verifiable voting, randomizing candidate order on ballot to encode votes, was first proposed by Ryan in [27] and elaborated in, for example, [23]. "Classical" Prêt à Voter is a polling station electronic voting scheme shown to be receipt-free. In this section we describe the key idea behind the design.

To facilitate our discussions, instead of denoting an encryption algorithm as $\mathsf{Enc}(m, pk)$, we use a slightly different notation $\mathsf{Enc}(pk, m, r)$, where pk is the public key, m is the message, and r is the randomness used in the encryption.

3.1 Prêt à Voter Overview

Here we explain the the key elements of Prêt à Voter , abstracting from details that are not relevant to the present anaysis.

Ballot Structure: We first explain the structure of the "Prêt à Voter " ballots. The ballot has a left hand side (LHS) with a randomly permuted list of candidates, and a right hand side (RHS) which carries an encryption of the order of the candidates in the LHS, usually referred to as *the onion* for historical reasons. Each ballot has a unique serial number (which could be a hash of the onion), (SN), for administrative purposes such as searching for the ballot on the bulletin board, etc. (See Fig. 3).

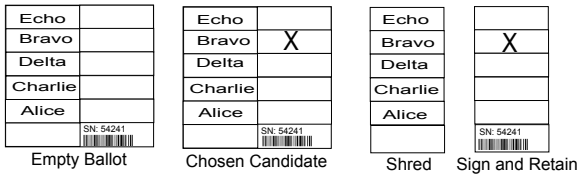

Figure 3: Prêt à Voter : The Ceremony

Overall System: Once registered in the polling station, the voter receives, via a confidential channel (e.g. Print–on–demand, sealed envelopes, etc), the ballot. The voter is offered the choice to cast the vote with this ballot or to audit it. If the chooses the former, she marks here preferences in the privacy of a voting booth. If the latter, officials or helpers are available to assist her in the ballot audit, after which she can obtain a fresh ballot.

In the booth the voter places a mark next to the name of the candidate she wants to vote for. She separates the RHS from LHS, shreds the LHS and takes the RHS to an official who scans and sends it to the tallying authority. A signed, franked copy of the RHS is given to the voter to keep. All receipts and audited ballots are published on a bulletin board. Voters can verify that their votes has been accurately entered in the tally by checking the onion, serial number and choice of index, against the published results on the bulletin board.

Tallying authorities compute the result of the elections and the result is published on the board. Figure 4 shows the ballots' cycle from the point its created to the point where a receipt or audited ballot is published on the bulletin board.

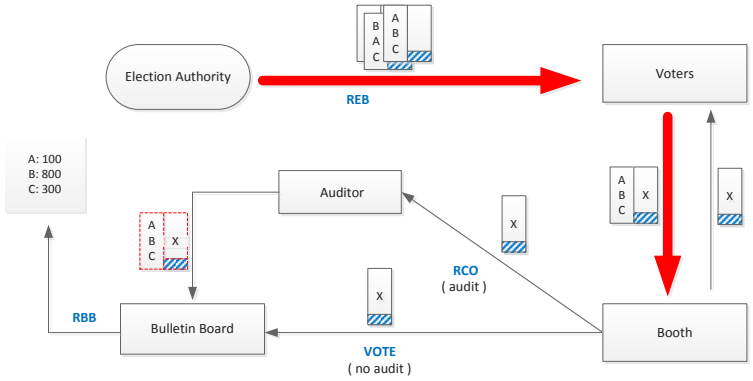

Figure 4: Prêt à Voter : The System

Random auditing the ballots is used in all versions of Prêt à Voter to ensure the well-formedness of ballot forms but differs slightly in style. The auditing procedure involves decrypting onions on selected ballot forms 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. An audited ballot can not be used for voting.

Tallying the result: The onions are used in the tabulation to interpret the voters mark on the scanned RHS, enabling the tallying authorities to count the votes. The details of the procedure varies in the different versions of Prêt à Voter [23,24,32]. Loosely speaking, the vote processing part of Prêt à Voter takes a set of encrypted votes (receipts) and outputs a set of unencrypted votes, but without allowing anyone (including those involved in the decryption) to perform end–to–end matching. This is done by mixing, decrypting, and tallying; where some variants of Prêt à Voter combine the mixing and decrypting procedure (using mix–nets), and other variants combine the decrypting and tallying phases (homomorphic tallying). Regardless of what vote processing method is used in Prêt à Voter the bulletin board will always have the serial number, onion and the index published.

To be able to provide a proof for the general case of Prêt à Voter we assume:

- Every ballot is identified by a unique serial number and two ballots should not have the same RHS.
- The voting takes place in a private booth in the supervised polling station.
- The ballots are distributed in a confidential way (e.g.post or print–on–demand).
- Counter-measures are in place to prevent any subliminal channels leaking information about the LHS via data on the RHS (e.g. distributed creation of encryptions to suppress kleptographic channels.)
- The attacker does not get access to ballots forms that are used to cast votes.
- The LHS of the ballot is shredded and destroyed in the booth.

- Audited ballots are not used for voting.
- The tabulation process reveals only the final tally information. Strictly speaking it will typically also provide proofs of correctness of these results, but these we assume to be zero–knowledge and hence revealing nothing further that could undermine ballot secrecy. We might use for example, a tabulation scheme proven secure in the UC paradigm, such as Wilkstroms, [30].

Different methodologies exist in literature to enforce these properties and counter measures can be found in the Prêt à Voter literature [23, 24, 32].

3.2 Construction of Ballots in the Versatile Prêt à Voter

We revisit the work [3–5, 32], the latest versions of Prêt à Voter , and adopt certain ideas used in their work to create the ballots. The ballots in these papers encoded the candidates as $\{M_0, \ldots, M_k\}$. The ballot generator creates a list of ciphertexts $\{\mathsf{Enc}(pk, M_0, 1), \ldots, \mathsf{Enc}(pk, M_k, 1)\}$. Anyone can verify that the ciphertexts are unique encryptions of the candidates since the randomization value is 1. Then the ciphertexts are re–encrypted and shuffled in a verifiable manner. The output is a permutation of $\{\mathsf{Enc}(pk, M_0, r_0), \mathsf{Enc}(pk, M_1, r_1), \mathsf{Enc}(pk, M_2, r_2), \ldots, \mathsf{Enc}(pk, M_k, r_k)\}$ that corresponds to LHS of the ballot. The proof of shuffle and the outputted list of ciphertext presents the onions. The papers used either exponential ElGamal [10] or Paillier [21].

3.3 The New Ballot Construction

In this section we show how one can have an IND-CCA2 ballot with minimum changes to the construct above. We assume the ballot generator has two sets of keys (pk_1, sk_1) and (pk_2, sk_2). The ballot generator creates a "shuffled" list of $\{(c_1, \grave{c}_1, \pi_1), \ldots, (c_k, \grave{c}_k, \pi_k)\}$, where $c_i = \mathsf{Enc}(pk_1, M_i, r_i)$, $\grave{c}_i = \mathsf{Enc}(pk_2, M_i, \grave{r}_i)$, π_i is a zero knowledge proof as described in §2.4 and M_i is one of the candidates codes. Furthermore, the ballot generator provides a verifiable proof of shuffle Υ on the input list of ciphertexts $\{\mathsf{Enc}(pk_1, M_0, 1), \ldots, \mathsf{Enc}(pk_1, M_k, 1)\}$ and the output of the shuffle are ciphertexts $\{c_1, \ldots, c_k\}$. Note that everyone can verify the inputted ciphertexts since the randomizations are 1, and no need to provide a proof of shuffle of $\{\grave{c}_1, \ldots, \grave{c}_k\}$ since $\pi_1, \ldots, \pi_k$ are provided. The onion of the ballot this time contains of the following: $\{(c_1, \grave{c}_1, \pi_1), \ldots, (c_k, \grave{c}_k, \pi_k)\}$ and Υ.

A ballot is well formed if an onion includes all ciphers of the different candidates (done by checking the verifiable shuffle proofs) and if the permutation in the onion corresponds to the LHS of the ballot (done by auditing). This relies on the soundness property of the employed zero knowledge proofs for Naor–Yung encryption, shown in Appendix B.3.

The voting procedure and processing remain the same as explained in §3.1. The fact that each candidate is encrypted with two different public keys is not entirely new for Prêt à Voter like schemes. Similar suggestions were proposed in literature [26] to provide Print–On–Demand for ballots and relax the trust assumptions based on the chain of custody[3].

[3] In Prêt à Voter the LHS is a plaintext which means that the chain of custody between the ballot generation and until the ballot reaches the voter should be trusted.

4 The Receipt Freeness Security model

The main advantage of using a cryptographic security model (in §4.3) over existing symbolic analysis of Prêt à Voter is to base the security of the scheme on exact computational hardness assumptions. A cryptographic proof should answer the question "How secure should the encryption scheme and proofs (used in creating the ballots) be, in order for Prêt à Voter to be receipt free?". Before introducing the security model of receipt freeness, we first give an overview in §4.1 on existing definitions and attacks related to confidentiality of the votes and coercion resistance. Our security model will capture some of them, while leaves the study of others as a future work.

4.1 Discussion on Confidentiality of Votes

The easiest way to preserve confidentiality of the vote is to have the voters encrypt the votes and then break the link between the voter and the votes by using mix–nets. Observing however that a coercer could demand that the voter reveal her private key or the randomness used for the encryption motivates stronger definitions that hold against such a coercer. In the context of Prêt à Voter , the encryptions are formed in advance and so the voter does not have access to such information as the randomization.

In literature certain subtle attacks were discovered and studied that relate to the confidentiality of a vote in an election scheme. A lot were addressed for Prêt à Voter by changing certain implementation details [26]. The following is a list:

- Auditing voted ballots: if an attacker is able to audit a ballot that is used to cast a vote he can violate its secrecy. Our model captures such attacks.
- Ballot dependence: if an attacker can construct a ballot from one or more cast ballots and inject it into the tabulation he may be able to obtain information about the cast ballots. Our model captures such attacks.
- Chain-voting. This is an attack to which a coercer obtains a blank ballot, marks it with his candidate and hands it to a voter with instructions to cast it and then pass him a fresh ballot. The coercer can continue the process indefinitely, marking each fresh ballot and obtaining another from the next voter. Our model does not capture such attacks.
- Randomization: if an attacker can instruct a voter before voting on some aspect of the receipt he may be able to prevent the voter from voting freely, even though he might not be able to determine the vote. In Prêt à Voter for example the coercer can instruct the voter to place her X in the first position. In the absence of counter-measures, this effectively randomizes the vote. Our model does not capture such attacks.
- Ballot signature ("Italian Attack"): If the options available to the voter as large, as is the case for example with Single Transferable Vote (STV) systems with a large number of candidates, it is possible for the coercer to require the voter to provide a unique "signature" on her ballot, e.g. a particular ranking of low ranked candidates. Our model does not capture such attacks.

In general, our model does not capture attacks, in which the attacker not only gets access to receipts but also interacts with the voters. This can be justified in terms of assumptions we make about the implementation of the scheme (See §3.1). These assumptions are reflected in the model in terms of restrictions that the adversary has

when interacting with the challenger. We argue that, for the purposes of defining receipt freeness for Prêt à Voter , our model embodies the most powerful attacker consistent with such reasonable assumptions. On the other hand it is not powerful enough to capture coercion resistance including randomization attacks, Italian attacks and chain voting. Elaborating the model for this purpose will be the subject of future work.

4.2 Oracles for the Adversary

The security model is defined using an attack game between a hypothetical adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ tries to win the game by issuing certain oracles to $\mathcal{C}$. Referring to Figure 4 which shows these oracles and what they represent in the real system, our security model has the following oracles.

- **Retrieve Empty Ballot (REB):** $\mathcal{A}$ sends a request to $\mathcal{C}$ for an empty ballot that he would like to get. $\mathcal{C}$ generates and returns an empty ballot (i.e. a ballot with both RHS and LHS).
- **Reveal Candidate Order (RCO):** $\mathcal{A}$ sends the RHS of the ballot used or not, and $\mathcal{C}$ responds with the candidate order on that ballot. This oracle reflects the audit capability that an adversary may have.
- **Vote:** $\mathcal{A}$ sends a vote (i.e a RHS of the ballot marked on the position of $\mathcal{A}$'s choice) to $\mathcal{C}$. Upon receiving a vote, $\mathcal{C}$ first validates it according to the procedure specified in the underlying voting scheme, and then puts it on the bulletin board if it is valid. Otherwise, $\mathcal{C}$ rejects the vote. It is worth noting that $\mathcal{A}$ can either use an empty ballot from $\mathcal{C}$ to construct a vote or forge a vote at its own interest.
- **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 added subsequently and $\mathcal{A}$ can query RBB again.

4.3 Receipt Freeness Attack Game

In the game, we assume that there are two candidates (i.e. Alice and Bob) for simplicity of explanation. Nevertheless, it can be easily extended to include more. Given the above oracles and the public information in the system, the goal of a receipt freeness adversary is to distinguish between two honestly generated receipts by telling which one is for Alice. Formally, the game is shown below.

In the attack game, trusted and compromised parties are each considered to be a single entity, i.e. the challenger $\mathcal{C}$ and adversary $\mathcal{A}$ respectively. As we focus here on proving receipt-freeness only, we assume that $\mathcal{A}$ is not interested in any other attacks. As long as the ballots used to query the oracles are not compromised, $\mathcal{A}$ should not be able to distinguish between the value of the votes of receipts i and j. Note that, if all voters have voted the same way, the confidentiality of the votes is by nature broken. Therefore, we require that the two receipts represent votes for different candidates.

We refer to the model above as EXP_{RF}, the probability of guessing the receipt in case of two candidates should be $\frac{1}{2}$. Receipt freeness for a two-candidate election is defined in Definition 4.1.

Definition 4.1 *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.*

- RF.Setup: $\mathcal{C}$ sets up the system by creating the private and public parameters, then sets up an empty bulletin board. The bulletin board is accessible to $\mathcal{A}$ as "read only". $\mathcal{C}$ sends the public parameters to $\mathcal{A}$.
- RF.Phase[I]: $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles.
- RF.Challenge: $\mathcal{A}$ asks for a challenge challenge and $\mathcal{C}$ returns the two receipts, one as a vote to Alice and the other as vote to Bob, randomly assigned. The ballots corresponding to the receipts should be generated by $\mathcal{C}$ and have not been queried in the REB, Vote, and RCO oracles. Let's assume the serial numbers of the two receipts are i and j
- RF.Phase[II]: $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles. The challenged receipts i and j can only be queried in RBB simultaneously.
- RF.Guess: $\mathcal{A}$ 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.

Figure 5: Receipt Freness Attack Game

For Prêt à Voter , the receipt freeness is about the ability to construct a proof to another party via using the receipts and the bulletin board information. The physical constraints (private booth and shredding LHS) in the real system are presented in the model by allowing the challenger to submit the challenged votes without the influence of the adversary while it gives $\mathcal{A}$ the receipts and read only access to BB. Relating the model to §4.1 auditing voted ballots is captured by restricting the RCO oracle where the receipts i and j are not allowed to be queried in RCO. Ballot dependence is captured because the votes submitted to RCO or Vote oracles can be constructed based on any existing ones (including the challenged ones).

5 The Proof of Receipt Freeness

In this section, we prove the following theorem for the new construction in section 3.3.

Theorem 5.1 *Given the proofs used in constructing ballots are zero knowledge, if there exists an Adversary $\mathcal{A}$ that can break the receipt freeness of Prêt à Voter then there exists a simulator $\mathcal{S}$ that breaks the IND-CCA2† security of Naor–Yung encryption.*

The intuition behind our proof is fairly straightforward. Assuming the existence of an adversary $\mathcal{A}$ who can win the receipt freeness game described in § 4 with a non-negligible advantage, then we can construct a simulator $\mathcal{S}$ which can win the IND–CCA2† security game of Naor–Yung scheme. Basically, the simulator $\mathcal{S}$ takes input from Naor–Yung challenger $\mathcal{C}$, who sets up the Naor–Yung Encryption scheme, and uses that input in playing the role of the receipt freeness challenger with $\mathcal{A}$ in the receipt freeness game.

Proof. Let's assume that $\mathcal{A}$'s advantage in attacking the receipt freeness game is ϵ^*, where the game is defined in Figure 5. The proof can be conducted with the standard game-hopping technique.

Consider a new game, denoted as Game1. In Game1, $\mathcal{S}$ plays the role of receipt freeness challenger and performs identically to what is required in Figure 5, except for the following: in the RF.Challenge phase, the shuffle proofs of the two receipts are removed. Note that the shuffle proof on a receipt proves that the two Naor–Yung ciphertexts are encryptions for Alice and Bob respectively. Let's assume that $\mathcal{A}$'s advantage in this game is ϵ. Given the fact that the shuffle proofs are zero knowledge, $\mathcal{A}$'s's advantage should be the same with/without the presence of the proofs, namely $|\epsilon - \epsilon^*|$ is negligible.

Consider a new game, denoted as Game2. In this game, $\mathcal{S}$ tries to win IND–CCA2† security game of Naor–Yung scheme (shown in Figure 2), by leveraging on $\mathcal{A}$'s responses. In more detail, $\mathcal{S}$ interacts with the Naor–Yung challenger $\mathcal{C}$ and $\mathcal{A}$ as follows.

- **Simulating the RF.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 and is initially empty. He also creates a private database of ballots DB that will contain $(SN, onion, candidates.order, vote)$. Initially that database is empty (it gets filled in RF.Phase I, II). He passes to $\mathcal{A}$ the values in $NY.pk$. See Figure 6.

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

Figure 6: Setup Phase

- **Simulating RF.Phase[I] and RF.Phase[II]**
  In these two phases, $\mathcal{A}$ queries the REB, RCO, RBB, and Vote oracles, and $\mathcal{S}$ answers as in Figure 7.

$\mathcal{A}$		$\mathcal{S}$		$\mathcal{C}$
Oracle REB:				
	$\xrightarrow{REB}$	Create $Ballot$;		
		Update DB with vote=null;		
	$\xleftarrow{response}$	response = empty ballot;		
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 RBB:				
	$\xrightarrow{RBB}$	Result = Tally BB		
	$\xleftarrow{Results}$			

Figure 7: Phase I and Phase II

The following explains the details of the simulation:

- REB: $\mathcal{S}$ receives a request for an empty 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 $null$. Note that SN is the unique serial number for the empty ballot.
- Vote: After receiving the query (.ie. an onion and index), $\mathcal{S}$ checks the ballot on the bulletin board if it has been used before. If so, he returns the used ballot to the adversary. Otherwise, he checks the $vote$ element in the database, if the ballot exists but the vote does not, he updates bulletin board and updates database to have $vote = c.name$. If the ballot does not exist in database then it must be created by the adversary, $\mathcal{S}$ queries the decryption oracle from $\mathcal{C}$ to get the information of the ballot and updates 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.
- RCO: $\mathcal{S}$ checks if the ballot is in the database to get the candidate's order otherwise query the decryption oracle, update the database then return the candidate order.
- RBB: $\mathcal{S}$ tallies the bulletin board and returns the result.

- **Simulating RF.Challenge and RF.Guess**
  In the Challenge and Guess phases, the interactions among different entities are summarised in Figure 8. Without loss of generality, we assume that the first element of each ballot is ticked, namely the candidate corresponding to the first ciphertext is voted and the receipt is a pair of Naor–Yung ciphertexts.

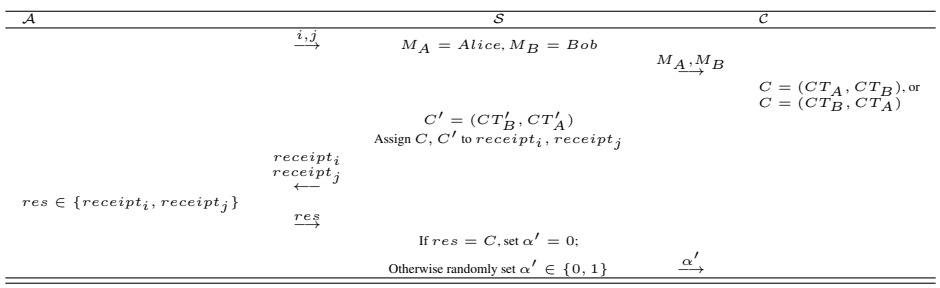

Figure 8: Challenge and Guess Phases

As shown in the figure, the essential strategy of $\mathcal{S}$ is to simulate the challenge (i.e. two receipts $\{receipt_i, receipt_j\}$) for $\mathcal{A}$ with two pieces of information: one is the Naor–Yung challenge C from $\mathcal{C}$ and the other is a Naor–Yung ciphertext pair C' generated by itself. With the response from $\mathcal{A}$, $\mathcal{S}$ can form a guess for $\mathcal{C}$'s random coin α. The details of RF.Challenge and RF.Guess are elaborated below.

1. $\mathcal{A}$ picks two serial numbers i and j, where the two ballots have not been queried before in REB, Vote and RCO and sends them to $\mathcal{S}$ [4].

[4]Recall from §3 the game restricts the two ballots i and j from being queried in oracles above because

2. $\mathcal{S}$ assigns $M_A = Alice$ and $M_B = Bob$ and sends them to $\mathcal{C}$.
3. $\mathcal{C}$ first computes Naor–Yung ciphertexts CT_A, CT_B for M_A, M_B respectively. $\mathcal{C}$ then tosses a coin $\alpha \in \{0,1\}$, and returns $C = (CT_A, CT_B)$ if $\alpha = 0$ and $C = (CT_B, CT_A)$ otherwise.
4. After receiving C, $\mathcal{S}$ generates $C' = (CT'_B, CT'_A)$, where CT'_A, CT'_B are two new Naor–Yung ciphertexts for M_A, M_B respectively. Based on C and C', $\mathcal{S}$ generates two receipts $\{receipt_i, receipt_j\}$ by flipping a coin $\beta \in \{0,1\}$. If $\beta = 0$, set $receipt_i = C$ and $receipt_j = C'$; otherwise, set $receipt_i = C'$ and $receipt_j = C$. Finally, $\mathcal{S}$ sends $\{receipt_i, receipt_j\}$ to $\mathcal{A}$ and adds them to the bulletin board.
5. $\mathcal{A}$ sends a receipt $res \in \{receipt_i, receipt_j\}$ back to indicate a vote for Alice.
   - When $\mathcal{C}$'s random coin is $\alpha = 0$, $\{receipt_i, receipt_j\}$ represent two receipts for Alice and Bob respectively and $\mathcal{S}$ is playing a faithful receipt freeness game with $\mathcal{A}$. Therefore, the probability that $\mathcal{A}$ can identify Alice's receipt is $\frac{1}{2} + \epsilon$.
   - When $\mathcal{C}$'s random coin is $\alpha = 1$, $\{receipt_i, receipt_j\}$ represent two receipts for Bob and $\mathcal{S}$ is not playing a faithful receipt freeness game with $\mathcal{A}$. Nevertheless, we can still ask $\mathcal{A}$ to pick one receipt res and send it back. Due to the fact that $receipt_i$ and $receipt_j$ have the identical distribution in this case, the probability that res equals to either of $\{C, C'\}$ is exactly $\frac{1}{2}$.
6. After receiving res from $\mathcal{A}$, $\mathcal{S}$ answers the challenge from the $\mathcal{C}$ as follows.
   - If the res is the Naor–Yung ciphertext pair C generated by $\mathcal{C}$, $\mathcal{S}$ outputs a guess $\alpha' = 0$.
   - If res is the Naor–Yung ciphertext pair C' generated by $\mathcal{S}$, $\mathcal{S}$ randomly selects α' from $\{0,1\}$ as a guess.

Depending on the value of α, various probabilities are summarised in Table 1.

$\alpha = \mathbf{0}$	$\alpha = \mathbf{1}$
$C = (CT_A, CT_B), C' = (CT'_B, CT'_A)$	$C = (CT_B, CT_A), C' = (CT'_B, CT'_A)\}$
$\Pr[res = (CT_A, CT_B)] = \frac{1}{2} + \epsilon$ $\Pr[res = (CT'_B, CT'_A)] = \frac{1}{2} - \epsilon$	$\Pr[res = (CT'_B, CT'_A)] = \frac{1}{2}$ $\Pr[res = (CT_B, CT_A)] = \frac{1}{2}$
$\Pr[\alpha' = \alpha \mid res = (CT_A, CT_B)] = 1$ $\Pr[\alpha' = \alpha \mid res = (CT'_B, CT'_A)] = \frac{1}{2}$	$\Pr[\alpha' = \alpha \mid res = (CT_B, CT_A)] = 0$ $\Pr[\alpha' = \alpha \mid res = (CT'_B, CT'_A)] = \frac{1}{2}$

Table 1: Probability Summary

In summary, $\mathcal{S}$'s advantage in winning the IND-CCA2† game against the Naor–Yung scheme is:

$$\varepsilon' = |\tfrac{1}{2}(\tfrac{1}{2} + \epsilon + \tfrac{1}{2}(\tfrac{1}{2} - \epsilon)) + \tfrac{1}{2}(\tfrac{1}{2} \cdot \tfrac{1}{2}) - \tfrac{1}{2}| = \tfrac{\varepsilon}{4}.$$

Based on Theorem 2.4 and Theorem 2.5, ε' is negligible so that ε is also negligible. Recall that $|\epsilon - \epsilon^*|$ is negligible. As a result, ϵ^* is negligible and the theorem follows. ■

they reveal the candidate order which contradicts with the challenge. In REB or RCO the order of the candidates get known directly. On the other hand, in Vote, the order can get to be known by querying these oracles as votes then tallying the bulletin board using RBB.

6 Conclusion

Prêt à Voter has been introduced in [23] as an end-to-end verifiable, receipt free voting scheme. Several formal method analysis exist in the literature to prove the properties of the scheme [24, 33]. In this paper we provide a proof of receipt freeness of Prêt à Voter using computational models. In order to do so, we defined a new security model that presents receipt freeness for Prêt à Voter . We show that creating the onions on the ballots with an IND–CCA2† secure encryption scheme is sufficient to achieve receipt freeness. We propose using Naor–Yung encryption for that purpose and that helps in maintaining the properties of one of the latest versions of Prêt à Voter [32]. We prove that the existence of an adversary that wins the receipt freeness security model implies the existence of a simulator that can break the IND-CCA2† security of Naor–Yung transformation. Extending this work to cover a larger family of voting schemes and to include stronger security notions such as receipt freeness is left for future work.

References

[1] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.

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

[3] Richard Buckland and Roland Wen. The future of e-voting in australia. *IEEE Security and Privacy*, 10(5):25–32, 2012.

[4] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. A supervised verifiable voting protocol for the victorian electoral commission. In *Electronic Voting*, pages 81–94, 2012.

[5] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. Using prêt à voter in victorian state elections. In *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'12, pages 1–1, Berkeley, CA, USA, 2012. USENIX Association.

[6] 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*, LNCS, pages 127–141. Springer.

[7] 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.

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

[9] 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.

[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1985.

[11] Benaloh Josh and Tuinstra Dwight. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, 1994.

[12] 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`.

[13] Ralf Küsters and Tomasz Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. *CoRR*, abs/0903.0802, 2009.

[14] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *CSF*, pages 122–136, 2010.

[15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Proving coercion-resistance of scantegrity ii. In *ICICS*, pages 281–295, 2010.

[16] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *IEEE Symposium on Security and Privacy*, pages 538–553, 2011.

[17] 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.

[18] 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.

[19] 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.

[20] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols Workshop*, pages 25–35, 1997.

[21] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

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

[23] Peter Y. A. Ryan. A variant of the chaum voter-verifiable scheme. In *Issues in the theory of security*, WITS, pages 81–88. ACM, 2005.

[24] 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.

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

[26] Peter Y.A. RYAN, David BISMARK, James HEATHER, Steve SCHNEIDER, and ZHE XIA. The Prêt à Voter Verifiable Election System. In *Transactions in Information Security and Forensics*. IEEE, 2009.

[27] P.Y.A. Ryan. A variant of the chaum voting scheme. Technical Report CS-TR-864, UNT, 2004.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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 Analysis of IND-CCA2†

Proof of Theorem 2.4. We first show that an IND-CCA2† secure scheme is also IND-CCA2 secure (namely the "if" part). Suppose that the scheme is not IND-CCA2 secure, then some attacker $\mathcal{A}$ has a non-negligible advantage in the game shown in Fig. 1. Then we construct an attacker $\mathcal{A}^\dagger$, which runs $\mathcal{A}$ as a subroutine to attack the encryption scheme in the game shown in Fig. 2. The attack is described in Fig. 9.

In this attack, $\mathcal{A}^\dagger$ can faithfully answer $\mathcal{A}$'s decryption queries as long as the two ciphertexts in E_d are not queried by $\mathcal{A}$. We take $d = 0$ as an example, where $E_0 = (\mathsf{Enc}(M_0, pk), \mathsf{Enc}(M_1, pk))$. In this case, $\mathsf{Enc}(M_0, pk)$ is not allowed to be the input of the decryption oracle in the game shown in Fig. 1 because it is the challenge, so that it will not be queried. We only need to show that $\mathcal{A}$ can only generate $\mathsf{Enc}(M_1, pk)$ with a negligible probability. This is a straightforward fact from the IND-CPA security

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker $\mathcal{A}^\dagger$ is given pk, and it sends pk to $\mathcal{A}$. If $\mathcal{A}$ issues a decryption query with C, $\mathcal{A}^\dagger$ submits C to the challenger. Once $\mathcal{A}^\dagger$ receives $\mathsf{Dec}(C, sk)$ from the challenger, it forwards this value to $\mathcal{A}$. When $\mathcal{A}$ submits M_0, M_1 for a challenge, $\mathcal{A}^\dagger$ send them to the challenger.
3. *Challenge*. Once receiving E_d from the challenger, $\mathcal{A}^\dagger$ sends the first ciphertext of E_d to $\mathcal{A}$. In more detail, if $d = 0$, then $\mathsf{Enc}(M_0, pk)$ is sent; otherwise $\mathsf{Enc}(M_1, pk)$ is sent.
4. *Phase 2*. The attacker $\mathcal{A}^\dagger$ deals with $\mathcal{A}$'s decryption queries in the same way as in *Phase 1*. If $\mathcal{A}$ terminates by outputting a guess b', then $\mathcal{A}^\dagger$ terminates by outputting a guess $d' = b'$.

Figure 9: IND-CCA2† Attack

property: if $\mathcal{A}$ can predicate the output of Enc for the input M_1, then it can trivially tell the ciphertext of M_1 from anything else and break the IND-CPA security. As a result, $\mathcal{A}^\dagger$ faithfully answer $\mathcal{A}$'s decryption queries! It is clear that $\mathcal{A}$ wins if and only if $\mathcal{A}^\dagger$ wins, so that $\mathcal{A}^\dagger$ has a non-negligible advantage. This conflicts with the assumption that the scheme is IND-CCA2† secure, hence, the scheme should be IND-CCA2 secure.

We next show that an IND-CCA2 secure scheme is also IND-CCA2† secure (namely the "only if" part). Suppose that the scheme is not IND-CCA2† secure, then some attacker $\mathcal{A}^\dagger$ has a non-negligible advantage ϵ in the game shown in Fig. 2. Then we construct an attacker $\mathcal{A}$, which runs $\mathcal{A}^\dagger$ as a subroutine to attack the encryption scheme in the game shown in Fig. 1. The attack is described in Fig. 10.

1. *Setup*. The challenger takes the security parameter λ as input, and runs KeyGen to generate (pk, sk).
2. *Phase 1*. The attacker $\mathcal{A}$ is given pk, and it sends pk to $\mathcal{A}^\dagger$. If $\mathcal{A}^\dagger$ issues a decryption query with C, $\mathcal{A}$ submits C to the challenger. Once $\mathcal{A}$ receives $\mathsf{Dec}(C, sk)$ from the challenger, it forwards this value to $\mathcal{A}^\dagger$. When $\mathcal{A}^\dagger$ submits M_0, M_1 for a challenge, $\mathcal{A}$ sends them to the challenger.
3. *Challenge*. Once receiving C_b from the challenger, $\mathcal{A}$ selects $t \in_R \{0, 1\}$ and sends $E_d = (C_b, \mathsf{Enc}(M_t, pk))$ to $\mathcal{A}^\dagger$. Note that, if $t \neq b$ then E_d is a valid challenge, otherwise it is not valid.
4. *Phase 2*. The attacker $\mathcal{A}$ deals with $\mathcal{A}^\dagger$'s decryption queries in the same way as in *Phase 1*. If $\mathcal{A}^\dagger$ terminates by outputting a guess d', then $\mathcal{A}^\dagger$ terminates by outputting a guess $b' = d'$.

Figure 10: IND-CCA2 Attack

In this attack, $\mathcal{A}$ can faithfully answer $\mathcal{A}^\dagger$ decryption queries but the challenge is faithfully generated only when $t \neq b$. Moreover, when $t \neq b$, $\mathcal{A}$ wins if and only if $\mathcal{A}^\dagger$ wins. Therefore, if $t \neq b$, $\mathcal{A}$'s advantage is ϵ. When $t = b$, suppose $\mathcal{A}$'s advantage is ϵ', which is a positive number and can be negligible with respect to the security parameter. As a result, $\mathcal{A}$'s overall advantage is $\frac{\epsilon+\epsilon'}{2}$, which is non-negligible given that ϵ is non-negligible. This conflicts with the assumption that the scheme is IND-CCA2 secure,

hence, the scheme should also be IND-CCA2† secure.

B Non-interactive Zero knowledge proofs

We 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.

B.1 Knowledge of discrete logs:

Proving knowledge of x, given h where $h \equiv g^x \bmod p$ [6, 7, 29].

- 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, $h \equiv g^x \bmod p$.

B.2 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 $k \equiv g^x \bmod p$ [8, 22].

- 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 proof (PEP).

B.3 Plaintext Equivalence Proof

This proof relies on Equality between discrete logs (see appendix B.2). Let $CT_1 = Enc(pk_1, m, \zeta_1) = (u_1, v_1) = (g^{\zeta_1}, {h_1}^{\zeta_1} g^m)$, $CT_2 = Enc(pk_2, m, \zeta_2) = (u_2, v_2) = (g^{\zeta_2}, {h_2}^{\zeta_2} g^m)$, where $h_1 = g^{sk_1}$ and $h_2 = g^{sk_2}$. The PEP is as follows:

- Compute $e_1 = {h_1}^{\zeta_2}$, and $e_2 = {h_2}^{\zeta_1}$
- Compute two Zero knowledge proofs of Equality between discrete logs. One between (u_1, e_2) and one between (u_2, e_1). This is to prove that e_1, and e_2 are well formed.
- Compute $e_3 = \frac{u_1}{u_2} = g^{\zeta_1 - \zeta_2}$
- Compute $e_4 = \frac{v_1}{v_2} \cdot {e_1}^{-1} \cdot e_2 = (h_1 \cdot h_2)^{\zeta_1 - \zeta_2}$

The PEP is a proof of knowledge of the equality of the exponent $\zeta_1 - \zeta_2$ in (e_3, e_4) to the bases $(g, h_1 \cdot h_2)$. Given we have a Fiat–Shamir proof one can add to the hash function used in appendix B.2 the order of the ciphertexts that we are testing. In other words, commit to the order of either (CT_1, CT_2) or (CT_2, CT_1) such that if an adversary manipulates the order the proof fails to verify.