Event Driven Private Counters

Eu-Jin Goh
Stanford University
eujin@cs.stanford.edu

Philippe Golle
Palo Alto Research Center
pgolle@parc.com

Abstract. We define and instantiate a cryptographic scheme called "private counters", which can be used in applications such as preferential voting to express and update preferences (or any secret) privately and non-interactively. A private counter consists of an encrypted value together with rules for updating that value if certain events occur. Updates are private: the rules do not reveal how the value of the counter is updated, nor even whether it is updated for a certain event. Updates are non-interactive: a counter can be updated without communicating with its creator. A private counter also contains an encrypted bit indicating if the current value in the counter is within a pre-specified range.
We also define a privacy model for private counters and prove that our construction satisfies this notion of privacy. As an application of private counters, we present an efficient protocol for preferential voting that hides the order in which voters rank candidates, and thus offers greater privacy guarantees than any other preferential voting scheme.

1 Introduction

There are many applications in which it is desirable to keep one's personal preferences private. For example, consider the Australian national election, which uses preferential voting. Preferential or instant runoff voting is an election scheme that favors the "most preferred" or "least disliked" candidate. In preferential voting, voters rank their candidates in order of preference. Vote tallying takes place in rounds. In each round, the number of first place votes are counted for all remaining candidates and if no candidate has obtained a majority of first place votes, the candidate with the lowest number of first place votes is eliminated. Ballots ranking the eliminated candidate in first place are given to the second place candidate in those ballots. Every voters' preferences must be made publicly available in an anonymous manner for universal verifiability; that is, anyone can verify that the tallying is done correctly. Unfortunately, revealing all the preferences allows a voter to easily "prove" to a vote buyer that she has given her vote to a specific candidate by submitting a pre-arranged unique permutation out of the $(n-1)!$ (n is the number of candidates) possible candidate preference permutations. Note that n need not be large — $n = 11$ already gives over three and a half million such permutations.

Ideally, we would like to keep the preferences of voters private to the extent that the correct outcome of the election can still be computed.

Our Contribution. In this paper, we define a cryptographic scheme called "private counters", which can be used in any application where participants wish to hide but yet need to update their preferences (or any secret) non-interactively. We then present an efficient instantiation of a private counter using semantically secure encryption schemes [17]. We also define a privacy model for a private counter, and use this model to relate the security of our private counter instantiation to the semantic security of the encryption schemes.

Using private counters, we develop a protocol to run preferential elections. In our election scheme, the preferences of voters are kept private throughout vote tabulation and are never revealed, which even a standard non-cryptographic preferential voting scheme cannot achieve. Our solution can also be used in a real world preferential election with physical voting stations for creating each voter's ballot to ensure privacy during vote tabulation. In addition, our election scheme provides voter privacy, robustness, and universal verifiability.

Our preferential election scheme has computational cost $O(nt^4)$ for n voters and t candidates, whereas the current best cryptographic solution for preferential voting without using mixnets has exponential cost $O(n(t! \log(n))^2)$ and it also reveals all the (unlinked) voter preferences for vote tabulation. Note that a mix network solution also reveals all the voter preferences.

We note that our election scheme requires voters to submit a zero-knowledge proof that the private counters that express their vote are well formed. For efficiency, we require that these proofs be non-interactive, which typically involves applying the Fiat-Shamir heuristic [14]. Hence, security of the voting scheme is shown only in the random oracle model.

Simple solutions that do not work. We further motivate our construction of private counters by discussing briefly two natural but unworkable approaches to preferential elections. Consider a preferential election with t candidates. Let E denote a semantically secure encryption scheme with an additive homomorphism.

Binary counter. In "yes/no" elections based on homomorphic encryption, a voter's ballot typically consists of a ciphertext C_i for each candidate i, where $C_i = E(1)$ for the candidate i for whom a vote is cast, and $C_j = E(0)$ for all other candidates $j \neq i$. These ciphertexts can be viewed as binary counters associated with the candidates. The encrypted votes can easily be tallied because E has an additive homomorphism.
This approach fails in a preferential election because ballots cannot be efficiently updated after one or more candidates are eliminated. Roughly speaking, the difficulty is that binary counters are essentially stateless (they encode only one bit of information), whereas updating counters requires keeping track of more state. For example, a ballot needs to encode enough information so that the ballot is transferred to the third most preferred candidate if the first and second most preferred candidate have been eliminated. The space cost required to update these binary counters non-interactively is exponential in t: a voter must give update instructions for all 2^t possible subsets of eliminated candidates. The space cost can be decreased if interaction

with the voters is allowed during vote tallying, which is undesirable for any reasonably sized election.

Stateful counters. Another approach is to associate with each candidate i an encryption $C_i = E(r_i)$ of her current rank r_i among the other candidates. This approach also requires a $O(t^2)$ matrix containing encryptions of 0 and 1; row i is used to update the counters when candidate i is eliminated. These ciphertexts allow efficient updates: when a candidate is eliminated, we decrease by one the rank of all the candidates ranked behind the eliminated candidate (that is, we multiply the corresponding rank ciphertexts by $E(-1)$) and leave unchanged the ranks of other candidates (that is, we multiply them by $E(0)$, which is indistinguishable from $E(-1)$). On the other hand, it does not seem possible to tally such stateful counters efficiently without also revealing the preferences of individual voters; recall that in preferential elections, a vote goes to a candidate if and only if that candidate is ranked first – other candidates do not receive "partial" credit based on their ranking.

Related Work. We first note that our notion of a private counter is different from that of a cryptographic counter defined by Katz et al. [20]. Among other differences, a cryptographic counter can be updated by anyone whereas a private counter can only be updated by an authority or group of authorities holding a secret key. In addition, a private counter has a test function that outputs an encrypted bit denoting whether the counter's current value belongs to a range of values; this test function is crucial for our applications.

Cryptographic "yes/no" election schemes were proposed by Benaloh [8, 6, 3] and such elections have since received much research interest [5, 25, 10, 11, 18, 15, 12, 2, 19]. It is easy to see that any mix network scheme [7, 25, 18, 19, 16] immediately gives a solution to a preferential election: the mix network mixes encrypted ballots containing preferences and all ballots are decrypted at the end of the mixing; the normal tallying for preferential voting takes place with the decrypted preferences. The disadvantage of a mix network solution is that the preferences for all voters are revealed after mixing. Although the preferences cannot be linked to individual voters, revealing all the preferences allows a voter to "prove" to a vote buyer that she has given her vote to a specific candidate by submitting a unique permutation.

The only cryptographic solution not using mix networks to preferential voting that we are aware of is by Aditya et al. [1]. Let n be the number of voters and t be the number of candidates. They propose a solution using Paillier encryption [22] that has communication cost $t! \log(n)$ bits per vote and a computation cost of $O(n(t! \log(n))^2)$ to decide the outcome of the election. This exponential inefficiency resulted in Aditya et al. recommending the use of mix networks for a preferential election. Furthermore, their solution is no better (in terms of privacy) than a mixnet solution in that it also reveals all the permutations at the end of the election. In this paper, we show that it is possible to have an efficient solution to preferential voting, and yet provide more privacy than a standard non-cryptographic solution (or a mixnet solution).

Notation. For the rest of the paper, we denote that x is a vector by writing $\overrightarrow{x}$. For a vector $\overrightarrow{x}$, $\overrightarrow{x}_i$ denotes the ith element in the vector. Similarly, for a matrix Y, $Y_{i,j}$ refers to the element in the ith row and jth column. If we have k multiple instances of an object Z, then we differentiate these instances with superscripts $Z^1, \ldots, Z^k$. We denote the cartesian product of k integer rings of order M (modulo M) $\mathbb{Z}_M \times \cdots \times \mathbb{Z}_M$ as $\mathbb{Z}_M^k$. If E is an encryption scheme, $E(X)$ denotes an encryption of X using E, and $E^{-1}(Y)$ denotes the decryption of ciphertext Y. Finally, we say that a function $f : \mathbb{Z} \to \mathbb{R}$ is *negligible* if for any positive $\alpha \in \mathbb{Z}$ we have $|f(x)| < 1/x^\alpha$ for sufficiently large x.

2 Private Counters With Encrypted Range Test

The following parameters define a private counter:

- An integer $M > 1$. We let $\mathbb{Z}_M = \{0, \ldots, M-1\}$ represent the integers modulo M. We call $\mathbb{Z}_M$ the domain of the private counter.
- A range $R \subseteq \mathbb{Z}_M$, and an initial value $v_0 \in \mathbb{Z}_M$.
- A set of events $S_1, \ldots, S_k$ and corresponding update values $u_1, \ldots, u_k \in \mathbb{Z}_M$.
- Two (possibly the same) semantically secure encryption schemes E and F, with corresponding public/private key pairs $(\mathcal{PK}_E, \mathcal{SK}_E)$ and $(\mathcal{PK}_F, \mathcal{SK}_F)$. Note that the choice of a security parameter for the counter is implicit in the choice of E and F.

These parameters define a private counter comprised of a state C, and three functions $\mathsf{Eval}, \mathsf{Test}$ and Apply where:

- C is the state of the private counter;
- the function $\mathsf{Eval}(C, \mathcal{SK}_F) = v \in \mathbb{Z}_M$ returns the current value of the counter;
- the function $\mathsf{Test}(C)$ returns $E(1)$ if $\mathsf{Eval}(C, \mathcal{SK}_F) \in R$ and $E(0)$ otherwise;
- the function $\mathsf{Apply}(C, S_i, \mathcal{SK}_F) = C'$ outputs the new counter state C' after event S_i;

and the following properties hold:

- if we denote C_0 the initial state of the counter, we have $\mathsf{Eval}(C_0, \mathcal{SK}_F) = v_0$;
- if $C' = \mathsf{Apply}(C, S_i, \mathcal{SK}_F)$, then $\mathsf{Eval}(C', \mathcal{SK}_F) = \mathsf{Eval}(C, \mathcal{SK}_F) + u_i \mod M$;
- the function Apply can be called at most once per event S_i (this restriction is perfectly natural for the applications we consider).

The function Eval plays no operational role in a private counter. It is introduced here only to define Test and Apply (later we also use Eval in proofs), but is never invoked directly. For that reason, we define a private counter as a triplet $(C, \mathsf{Test}, \mathsf{Apply})$, leaving out Eval.

Extension. We can define more general counters that can handle tests of subset membership instead of just ranges. Since our applications do not require such general counters, we will not consider them further.

2.1 Privacy Model

Informally, a private counter should reveal no information about either its initial value or the update values associated with events. Note that these two properties imply that the subsequent value of the counter after one or several invocations of Apply remains private. We formally define privacy with the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Privacy Game 0

Setup: $\mathcal{C}$ generates public/private key pairs $(\mathcal{PK}_E, \mathcal{SK}_E)$ and $(\mathcal{PK}_F, \mathcal{SK}_F)$ for encryption schemes E and F, and also chooses a domain $\mathbb{Z}_M$ and a set of events $S_1, \ldots, S_k$. $\mathcal{C}$ gives $\mathcal{A}$ the public keys $\mathcal{PK}_E, \mathcal{PK}_F$, together with the domain and set of events. $\mathcal{A}$ outputs the range $R \subseteq \mathbb{Z}_M$, together with two initial values $v^*, v' \in \mathbb{Z}_M$ and two sets of corresponding update values $\overrightarrow{u}^*, \overrightarrow{u}' \in \mathbb{Z}_M^k$ for the k events.

Challenge: $\mathcal{C}$ flips a random bit b. $\mathcal{C}$ constructs the challenge private counter $(C_b, \mathsf{Test}, \mathsf{Apply})$ from $\mathcal{A}$'s parameters in the following way — If $b = 0$, $\mathcal{C}$ constructs a private counter C_0 using initial value v^* and update values $\overrightarrow{u}^*$; if $b = 1$, $\mathcal{C}$ constructs private counter C_1 with initial value v' and update values $\overrightarrow{u}'$.

Queries: $\mathcal{A}$ can request invocations to the function Apply from $\mathcal{C}$.

Output: $\mathcal{A}$ outputs its guess g for the bit b.

We say that $\mathcal{A}$ wins privacy game 0 if $\mathcal{A}$ guesses bit b correctly.

Definition 1. *A counter scheme is private according to game 0 if all polynomial (in security parameter t) time algorithms win game 0 only with negligible advantage* $\mathtt{Adv}(t) = |\Pr[g = b] - 1/2|$.

We use sets of counters in our applications so we extend privacy game 0 to multiple counters and denote the extended game as privacy game 1. Extending game 0 is straightforward and we give a precise definition in Appendix A. Privacy game 1 allows us to prove that we can use a set of private counters simultaneously while preserving privacy of individual counters; the proposition and proof is also found in Appendix A.

Note. The privacy requirements for our applications may appear different than the definition given by privacy game 0. For example, an adversary in an application may perform actions that are not described in privacy game 0 such as requesting for decryptions of the output of Test. In later sections describing each application, we will define precisely their privacy requirements and then show that the privacy definition given by game 0 is sufficient.

2.2 Construction

We present a private counter construction with domain $\mathbb{Z}_M$, subset $R \subseteq \mathbb{Z}_M$, initial value $v_0 \in \mathbb{Z}_M$ and a set of k events $S_1, \ldots, S_k$ with corresponding update values $u_1, \ldots, u_k \in \mathbb{Z}_M$. Furthermore, we restrict the domain $\mathbb{Z}_M$ to be at most

polynomial in size. Let E denote any semantically secure encryption scheme such as ElGamal [13] or Pailler [22], and let F be a semantically secure encryption scheme with an additive homomorphism modulo M such as Naccache-Stern [21] or Benaloh [4].

Counter State. The counter state consists of three parts: a $(k+1)$-by-M matrix of ciphertexts called Q, a pointer p that points to an element of the matrix Q, and two vectors of ciphertexts called $\overrightarrow{u}$ and $\overrightarrow{a}$. The matrices Q and two vectors $\overrightarrow{a}, \overrightarrow{u}$ are fixed and the function Apply only affects the value of the pointer p. The matrix Q, pointer p, and vectors $\overrightarrow{a}, \overrightarrow{u}$ are defined as follows:

Matrix Q**.** We first define a vector $\overrightarrow{w} = (w_0, \ldots, w_{M-1})$: let $w_j = E(1)$ if $j \in R$ and $w_j = E(0)$ if $j \notin R$. We now define the $(k+1)$-by-M matrix Q using $\overrightarrow{w}$.
Let $Q^0, \ldots, Q^k$ denote the rows of Q. Let $a_0, \ldots, a_k$ be $k+1$ random values chosen uniformly independently at random from $\mathbb{Z}_M$. For $i = 0, \ldots, k$, we define the row Q^i as the image of the vector $\overrightarrow{w}$ cyclically shifted a_i times to the right. That is, if we let $Q_{i,j}$ denote the element of Q in row $Q^i \in \{0, \ldots, k\}$, column $j \in \{0, \ldots, M-1\}$, we have $Q_{i,j} = w_{j-a_i}$, where the subscript $j - a_i$ is computed modulo M.

Pointer p**.** The pointer is a pair of integers $p = (i, j)$, where $i \in [0, k]$ and $j \in [0, M-1]$, that refer to ciphertext $Q_{i,j}$ in matrix Q. The initial state of the counter is defined as $p = (0, a_0 + v_0)$.

Vectors $\overrightarrow{a}, \overrightarrow{u}$**.** Vector $\overrightarrow{a}$ contains $k+1$ ciphertexts $F(a_0), \ldots, F(a_k)$, which are the encryptions of the $k+1$ random values $a_0, \ldots, a_k$ chosen for matrix Q. Vector $\overrightarrow{u}$ contains k ciphertexts $F(u_1), \ldots, F(u_k)$, which are the encryptions of the k update values $u_1, \ldots, u_k$.

Only the public key for E is needed to construct Q, and only the public key for F is required to build $\overrightarrow{a}$ and $\overrightarrow{u}$.

Computing Eval**.** Recall that the function Eval plays no operational role in a counter. Nevertheless, we describe how to compute Eval to help the reader understand the intuition behind our construction. Let (i, j) be the current value of the pointer p. $\mathsf{Eval}(C, \mathcal{SK}_F)$ returns the current value of the counter as the integer $(j - a_i) \mod M$.

Computing Test**.** Let (i, j) be the pointer's current value. $\mathsf{Test}(C)$ returns the ciphertext $Q_{i,j}$.

Computing Apply**.** We show how to compute the function $\mathsf{Apply}(C, S_l, \mathcal{SK}_F)$. Let $p = (i, j)$ be the current value of the pointer. Compute the ciphertext $F(a_l - a_i + u_l)$ by using the additive homomorphism of F on the appropriate ciphertexts from $\overrightarrow{a}$ and $\overrightarrow{u}$. Let d be the decryption of the ciphertext $F(a_l - a_i + u_l)$. $\mathsf{Apply}(C, S_l, \mathcal{SK}_F)$ outputs the new pointer $p' = (l, j + d)$ where the value $j + d$ is computed modulo M.

Privacy. Our counter construction is only private according to the privacy game 0 if the function Apply is never called twice for the same event. We note that

our voting application always satisfies this condition. Furthermore, the function Apply takes as input the secret key $\mathcal{SK}_F$, which lets the owner(s) of $\mathcal{SK}_F$ enforce this condition. We give a detailed proof of privacy in Section 2.3.

Cost. The size of our counter is dominated by $O(kM)$ ciphertexts from E and $O(k)$ ciphertexts from F. The computational cost of building a private counter is dominated by the cost of creating the ciphertexts. Computing the function Apply requires one decryption of F.

2.3 Proof of Privacy

We now prove that the construction of Section 2.2 is private provided the encryption schemes E and F are semantically secure and the function Apply is never called twice for the same event.

Recall that semantic security for an encryption scheme is defined as a game where the challenger $\mathcal{C}$ first provides the public parameters to the adversary $\mathcal{A}$, upon which $\mathcal{A}$ chooses and sends two equal length messages M_0, M_1 back to $\mathcal{C}$. $\mathcal{C}$ then chooses one of the messages M_b and returns the encryption of M_b to $\mathcal{A}$. The goal of the adversary is to guess the bit b. In our security proof, we use a variant of the semantic security game where the challenger returns both $E_0 = E(M_b)$ and $E_1 = E(M_{1-b})$ to the adversary. It is easy to see that this variant is equivalent (with a factor of two loss in the security reduction) to the standard semantic security game.

In the privacy game, recall that the adversary outputs two sets of initial values and update values $v^*, u_1^*, \ldots, u_k^*$ and $v', u_1', \ldots, u_k'$ as the choice for its challenge. The main difficulty in the security proof is in embedding the semantic security challenge ciphertexts E_b, E_{1-b} into the private counter's matrix Q so that if $b = 0$, the matrix Q represents initial value v, and if $b = 1$, the matrix Q represents initial value v'. Similarly, we have to embed the challenge ciphertexts F_b, F_{1-b} into the private counter's vector $\overrightarrow{u}$ so that if $b = 0$, vector $\overrightarrow{u}$ contains $F(u_1^*), \ldots, F(u_k^*)$, and $F(u_1'), \ldots, F(u_k')$ otherwise.

Proposition 1. *If the encryption schemes E and F are both semantically secure, the counter of Section 2.2 is private according to privacy game 0.*

Proof. We prove the proposition using its contrapositive. Suppose the counter of Section 2.2 is not private. Then there exists an algorithm $\mathcal{A}$ that wins the privacy game with non-negligible advantage; that is, $\mathcal{A}$ non-trivially distinguishes between a private counter with initial value v^* with update values $u_1^*, \ldots, u_k^*$ and a private counter with initial value v' with update values $u_1', \ldots, u_k'$. A standard hybrid argument shows that $\mathcal{A}$ can distinguish between two private counters with non-negligible advantage when the two counters have either —

Case 1: different initial values ($v^* \neq v'$) but the same update values ($u_i^* = u_i'$ for $1 \leq i \leq k$).

Case 2: the same initial values ($v^* = v'$) but different update values ($u_i^* \neq u_i'$ for at least one i where $1 \leq i \leq k$).

Case 1 implies that $\mathcal{A}$ distinguishes between two private counters based solely on the initial value and case 2 implies that $\mathcal{A}$ distinguishes based solely on the update values. If case 1 holds, then we build an algorithm $\mathcal{B}_1$ that breaks E. If case 2 holds, then we build an algorithm $\mathcal{B}_2$ that breaks F. Recall that F has an additive homomorphism modulo k.

Algorithm $\mathcal{B}_1$. We define an algorithm $\mathcal{B}_1$ that uses $\mathcal{A}$ to break the semantic security of E with non-negligible advantage. Algorithm $\mathcal{B}_1$ simulates $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{B}_1$ is given the encryption scheme E with the public key $\mathcal{PK}_E$ for a security parameter t. $\mathcal{B}_1$ generates the key pair $(\mathcal{PK}_F, \mathcal{SK}_F)$ for encryption scheme F. $\mathcal{B}_1$ begins by choosing two plaintexts $M_0 = 0$ and $M_1 = 1$ and receives as its challenge two ciphertexts $E_0 = E(M_b)$ and $E_1 = E(M_{1-b})$ for a random bit b. The goal of $\mathcal{B}_1$ is to guess the bit b.

$\mathcal{B}_1$ runs $\mathcal{A}$ with initial input the public keys $\mathcal{PK}_E, \mathcal{PK}_F$, an arbitrarily chosen domain $\mathbb{Z}_M$ (where M is polynomially large), and a set of events $S_1, \ldots, S_k$. In return, $\mathcal{A}$ outputs the range $R \subseteq \mathbb{Z}_M$, together with two initial values $v^*, v' \in \mathbb{Z}_M$ where $v^* \neq v'$ and two sets of update values $u_1^*, \ldots, u_k^* \in \mathbb{Z}_M$ and $u_1', \ldots, u_k'$ for the events.

Challenge: $\mathcal{B}_1$ constructs a private counter starting with matrix Q. We define two vectors:

1. $\overrightarrow{w}^* = (w_0^*, \ldots, w_{M-1}^*)$ where $w_j^* = 1$ if $j \in R$, and $w_j^* = 0$ if $j \notin R$, and
2. $\overrightarrow{w}' = (w_0', \ldots, w_{M-1}') = (w_{v'-v^*}^*, w_{v'-v^*+1}^*, \ldots, w_{M-1}^*, w_0^*, \ldots, w_{v^*-v'-1}^*)$ where the subscripts are computed modulo M. Note that $\overrightarrow{w}'$ is $\overrightarrow{w}^*$ cyclicly shifted by $v^* - v'$ (a negative value results in a right shift).

We want to construct the vector $\overrightarrow{w} = (w_0, \ldots, w_{M-1})$ with the following property: if $b = 0$, then $\overrightarrow{w}$ is the encryption of $\overrightarrow{w}^*$ and is defined exactly as described in Section 2.2 with domain $\mathbb{Z}_M$ and subset $R \subseteq \mathbb{Z}_M$; if $b = 1$, then $\overrightarrow{w}$ is the encryption of $\overrightarrow{w}'$. To obtain this property, vector $\overrightarrow{w}$ is built from $\overrightarrow{w}^*$ and $\overrightarrow{w}'$ as follows:

- If $w_j^* = 0$ and $w_j' = 0$, we let $w_j = E(0)$.
- If $w_j^* = 0$ and $w_j' = 1$, we let $w_j = E_0$.
- If $w_j^* = 1$ and $w_j' = 1$, we let $w_j = E(1)$.
- If $w_j^* = 1$ and $w_j' = 0$, we let $w_j = E_1$.

The $(k+1)$-by-M matrix Q is constructed exactly as described in Section 2.2 with our vector $\overrightarrow{w}$ and a set of random values $a_0, \ldots, a_k$. The vector $\overrightarrow{a}$ is built as $(F(a_0), \ldots, F(a_k))$ and initial value of pointer p is set to $(0, a_0 + v^*)$. To construct vector $\overrightarrow{u}$, $\mathcal{B}_1$ flips a coin and if 0 uses $u_1^*, \ldots, u_k^*$ to build $\overrightarrow{u}$, and if 1 uses $u_1', \ldots, u_k'$ instead.

$\mathcal{B}_1$ gives the resulting counter C to $\mathcal{A}$. Note that if $b = 0$, $\mathcal{A}$ receives a counter for initial value v^*, whereas if $b = 1$, $\mathcal{A}$ receives a counter for initial value v'.

Queries: $\mathcal{B}_1$ can compute Apply for $\mathcal{A}$ because $\mathcal{B}_1$ knows the values $a_0, \ldots, a_k$ and also $u_1^*, \ldots, u_k^*$ (respectively $u_1', \ldots, u_k'$).

Output: $\mathcal{B}_1$ outputs $g = 0$ if $\mathcal{A}$ guesses that C has initial value v^*. Otherwise, $\mathcal{B}_1$ outputs $g = 1$.

With probability 1/2, $\mathcal{B}_1$ chooses the right set of update values for vector $\overrightarrow{u}$ and the counter is well formed. It follows directly that $\mathcal{B}_1$ wins the semantic security game with non-negligible advantage.

Algorithm $\mathcal{B}_2$. We define an algorithm $\mathcal{B}_2$ that uses $\mathcal{A}$ to break the semantic security of F with non-negligible advantage. Algorithm $\mathcal{B}_2$ simulates $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{B}_2$ is given the encryption scheme F with the public key $\mathcal{PK}_F$ for a security parameter t. $\mathcal{B}_2$ generates the key pair $(\mathcal{PK}_E, \mathcal{SK}_E)$ for encryption scheme E. $\mathcal{B}_2$ begins by choosing two plaintexts $M_0 = 0$ and $M_1 = 1$ and receives as its challenge two ciphertexts $F_0 = F(M_b)$ and $F_1 = F(M_{1-b})$ for a random bit b. The goal of $\mathcal{B}_2$ is to guess the bit b. The rest of the Setup phase is identical to that for algorithm $\mathcal{B}_1$.

Challenge: $\mathcal{B}_2$ constructs a private counter as follows. The $(k+1)$-by-M matrix Q and vector $\overrightarrow{a}$ is constructed exactly as described in Section 2.2. To construct pointer p, $\mathcal{B}_2$ flips a coin and if 0 uses initial value v^* to build p, and if 1 uses initial value v' instead. The vector of encrypted update values $\overrightarrow{u} = (F(u_1), \ldots, F(u_k))$ is created from $u_1^*, \ldots, u_k^*$ and $u_1', \ldots, u_k'$ as follows: for all $1 \leq i \leq k$,

1. if $u_i^* = u_i'$, then $F(u_i) = F(u_i^*) = F(u_i')$.
2. if $u_i^* < u_i'$, then $F(u_i) = F(u_i^*) \cdot F_0^{u_i' - u_i^*} = F(u_i^* + b(u_i' - u_i^*))$.
3. if $u_i^* > u_i'$, then $F(u_i) = F(u_i') \cdot F_i^{u_i^* - u_i'} = F(u_i' + (i-b)(u_i^* - u_i'))$.

If $b = 0$, then $\overrightarrow{u}$ is the update vector created using $u_1^*, \ldots, u_k^*$. If $b = 1$, then $\overrightarrow{u}$ is update vector created using $u_1', \ldots, u_k'$. Note that the update vector $\overrightarrow{u}$ is computable because F has an additive homomorphism modulo k, and also because $u_i^*, u_i' \in \mathbb{Z}_M$ and $\mathbb{Z}_M$ is polynomial in size. Finally, $\mathcal{B}_2$ gives the resulting counter to $\mathcal{A}$.

Queries: Before any Apply queries are answered, $\mathcal{B}_2$ flips a coin and if 0 uses $u_1^*, \ldots, u_k^*$ to answer Apply queries, otherwise $\mathcal{B}_2$ uses $u_1', \ldots, u_k'$ instead. With this guess, $\mathcal{B}_2$ can answer Apply queries because $\mathcal{B}_2$ generates (and knows) $a_0, \ldots, a_k$.

Output: Algorithm $\mathcal{B}_2$ outputs $g = 0$ if $\mathcal{A}$ guesses that the counter contains the update values $u_1^*, \ldots, u_k^*$. Otherwise, $\mathcal{B}_2$ outputs $g = 1$.

With probability 1/2, $\mathcal{B}_2$ uses the correct set of update values to answer Apply queries, in which case, $\mathcal{B}_2$ wins the semantic security game for F with non-negligible probability. □

3 Preferential Voting

In this section, we give a cryptographic solution to preferential voting using our private counter construction. The participants of the election are:

1. n voters labelled $b_1, \ldots, b_n$.
2. t candidates standing for election labelled $x_1, \ldots, x_t$.

3. a number of election authorities that collect the votes, verify them, and collectively compute the result of the election. These election authorities share a single public key but the corresponding private key is shared among all of them. Encryptions are performed with the public key of the election authority but (threshold) decryption requires the consent of a quorum.

In voting, a voter's preferences must remain anonymous and her current first place candidate must never be revealed during vote tabulation. Despite this restriction, the election authorities must 1) tally up votes for each candidate, and 2) verify that ballots are valid by ensuring that a ballot has exactly one first place candidate and that preferences do not change arbitrarily from round to round.

Setup. The election authorities jointly generate the public/private key pair using a threshold protocol and publish the public parameters. For preferential voting, we require that E is the Paillier encryption scheme [22], which has an additive homomorphism. In addition, E should be a threshold version of the Paillier encryption scheme [15, 12]. The encryption scheme F can be any scheme with an additive homomorphism modulo t such as Naccache-Stern [21] and Benaloh [4].

Vote Creation. Each voter ranks the candidates in decreasing order of preference. For example, if a voter ranks 3 candidates in the order (x_2, x_3, x_1) where candidate x_2 is the first choice and candidate x_1 the last, we say that candidate x_2 has rank 0, candidate x_3 has rank 1, and candidate x_1 has rank 2. Note that the rank of candidate x_i is equal to the number of candidate ranked ahead of x_i.

Before describing how votes are created, we explore how eliminating a candidate affects the current preferences of voter b_i. Suppose that candidate x_1 has been eliminated. If b_i ranked x_1 ahead of x_2, the rank of x_2 should now be decreased by 1, moving it closer to first place. If x_1 was ranked behind x_2, then the rank of x_2 is unaffected by the removal of x_1. Note that this statement holds true regardless of the number of candidates (up to $t-2$) that have been eliminated so far. Therefore, the change in rank of x_2 when x_1 is eliminated depends only on whether x_2 ranked ahead or behind x_1 in the initial ranking of b_i.

Voter b_i creates her vote as follows. The vote consists of t private counters $P^{b_i,x_1}, \ldots, P^{b_i,x_t}$ (one counter for each candidate x_j where $j \in [1,t]$), together with zero knowledge proofs that these counters are well-formed (see Section 3.1). The domain D of each private counter is $D = [0, t-1)$ (the range of possible ranks) and the range R is $0 \in D$. In private counter P^{b_i,x_j}:

1. The initial value v is the initial rank assigned to candidate x_j by voter b_i.
2. Events $S_1, \ldots, S_t$ are the events that candidate x_l for $l \in [1,t]$ is eliminated.
3. The update value u_k associated with event S_k for $k \in [1,t]$ is $u_k = 0$ if voter b_i ranks x_k with a higher rank than x_j, and $u_k = -1$ if x_k has a lower rank than x_j.

Note that the the number of update values that are -1 is equal to the initial rank of candidate x_j, since the rank denotes the number of candidates preferred to x_j. Thus when a counter reaches 0 (first place choice), it can go no lower.

Vote Checking. The election authority checks that all the votes are well-formed (see Section 3.1), and discards invalid votes.

Vote Tallying. Recall that Test returns $E(1)$ if candidate x_j is the first place candidate and $E(0)$ otherwise. During each round of vote tallying, the election authorities compute the encrypted tally of first place votes for each candidate x_j as $\prod_{i=1}^{n} \mathsf{Test}(P_{b_i,x_j})$. The tally for each candidate is decrypted, requiring the consent of a quorum of private key holders. Note that since Test can be computed publicly, an honest election authority participates in the threshold decryption of the tally only if it is correctly computed.

Vote Update. If no candidate wins a majority of votes, the candidate with the fewest number of votes (say, candidate x_k) is eliminated. Voters who ranked the eliminated candidate x_k in first place now have their vote transferred to their second most preferred candidate in subsequent rounds. To do so, the election authorities update every voter's private counters to reflect their new first place preferences. The election authorities:

1. remove the k-th private counter from all votes; that is, remove counters P^{b_i,x_k} for $i \in [1, n]$.
2. invoke Apply on every vote's $t-1$ remaining private counters with the event that candidate x_k is removed. That is, for voter b_i where $i \in [1, n], i \neq k$, the election authorities invoke $\mathsf{Apply}(P^{b_i,x_j}, S_k, \mathcal{SK}_F)$ for $j \in [1, t]$. Note that no single election authority possesses $\mathcal{SK}_F$ and a quorum must be obtained for the necessary threshold decryptions for the Apply function.

The vote tallying, updating, and verifying process continues with successively less candidates until a candidate wins a majority of first place votes.

Cost. Each vote contains t private counters and so the space cost is $O(t^3)$ ciphertexts; as we will see in the next section, the space cost of the proofs is $O(t^4)$ ciphertexts. The computation required to create a vote is dominated by the cost of $O(t^4)$ encryptions for the proofs. Verifying a single vote costs $O(t^4)$. Tallying the first place votes for a single candidate requires one decryption. Updating a vote after each candidate is eliminated requires t decryptions. In summary, the election authority performs $O(nt^4)$ decryptions to compute the outcome of the election.

Security. During vote tabulation, the outputs of the function Test on the private counters in each ballot for all voters are tallied and decrypted. The adversary thus learns the decryption of the function Test "in aggregate". Informally, as long as $\mathcal{A}$ controls no more than a small fraction of the total number of voters, the aggregate tally reveals little about the individual vote of any voter. We note that every voting scheme that tallies votes using a homomorphic encryption scheme à la Cramer et al. [10, 11] has the same weakness.

Assuming that decryption of the aggregate counters is safe, the privacy of each voter's ballot follows directly from the privacy guaranteed by Proposition 1. That is, a voter's preferences are never revealed throughout vote submission and tabulation (even to the election authority). In preferential voting, a voter

can submit a unique permutation of preferences (which is revealed for universal verifiability) to "prove" how she voted. Non-cryptographic preferential voting and preferential voting using mix networks cannot prevent such privacy leaks but our scheme can because each voter's preferences are never revealed. Furthermore, the election is universally verifiable and anyone can verify that the submitted votes are valid and that the tallies every round are correctly computed. Lastly, the quorum of election authorities ensures that the voting scheme is robust, provided no more than a fraction of them are malicious.

3.1 Proving that a Vote is Valid

In many electronic election schemes, the voter must attach with her ballot, a proof (typically zero-knowledge or witness indistinguishable) that the ballot is correctly formed. For example, in a yes/no election, the voter must prove that the ballot really is an encryption of 0 or 1. Otherwise, the tally may be corrupted by a voter sending an encryption of an arbitrary value.

Efficient interactive zero knowledge proofs of bit encryption can be constructed for well known homomorphic encryption schemes such as Paillier [22, 12, 2] and Benaloh [4, 11]. These proofs are made non-interactive by applying the Fiat-Shamir heuristic, which replaces communication with an access to a random oracle [14]. In practice, the random oracle is replaced by a cryptographic hash function. Security holds in the random oracle model and not in the standard model [23]. Instead of applying the Fiat-Shamir heuristic, we could instead use a trusted source of random bits such as a beacon [24] so as to obtain security in the standard model, but the resulting constructions are less efficient.

In our election scheme, a voter must prove to the election authority in non-interactive zero-knowledge (NIZK) that the t counters expressing her vote are well-formed. Specifically, the voter must prove 1) that the counters only express one first place vote at any given time, and 2) that the transfer of votes as candidates are eliminated proceeds according to a fixed initial ranking of candidates; that is, a vote must always be transferred to the next most preferred candidate among those remaining.

We require NIZK proofs that the decryption $E^{-1}(C)$ (resp. $F^{-1}(C)$) of a ciphertext C lies within a given set of messages $m_0, \ldots, m_t$; we denote such a proof as $\mathsf{NIZKP}\left\{E^{-1}(C) \in \{m_0, \ldots, m_t\}\right\}$ (resp. with F). The size and computational cost to create and verify such a proof is linear in t (the size of the set) [12, 2, 11]. These proofs can also be combined conjunctively and disjunctively using standard techniques [9, 26].

Recall that we denote by t the number of candidates. A vote consists of t counters, with matrices $Q^1, \ldots, Q^t$, initial pointers $p^1, \ldots, p^t$, cyclic shift vectors $\overrightarrow{a}^1, \ldots, \overrightarrow{a}^t$ and update vectors $\overrightarrow{u}^1, \ldots, \overrightarrow{u}^t$. To prove that these counters are well-formed, a voter does the following:

1. The voter commits to her initial ranking of candidates. This commitment takes the form of t ciphertexts, $C_1, \ldots, C_t$, where C_i is an encryption with F of the initial rank of candidate x_i.

2. The voter proves that the commitment given in step 1 is well-formed; that is, the voter proves that the ciphertexts $C_1, \ldots, C_t$ are encryptions of the values $0, \ldots, t-1$ permuted in a random order. This property is proved by showing that for all $i \in \{0, \ldots, t-1\}$, there exists j such that $C_j = F(i)$. Formally, the voter proves for all $i \in \{0, \ldots, t-1\}$ that $\bigvee_{j=1,\ldots,t} \mathsf{NIZKP}\left\{F^{-1}(C_j) \in \{i\}\right\}$.
3. The voter proves that each matrix Q^k for $k \in \{1, \ldots, t\}$ is well-formed:
   - for all $k \in \{1, \ldots, t\}$, $i \in \{0, \ldots, t\}$, and $j \in \{1, \ldots, t\}$, the entry $Q^k_{i,j}$ of matrix Q^k is an encryption of either 0 or 1. Formally, the voter creates $\mathsf{NIZKP}\left\{E^{-1}(Q^k_{i,j}) \in \{0,1\}\right\}$.
   - for all $k \in \{1, \ldots, t\}$ and for all $i \in \{0, \ldots, t\}$, there is one and only one entry in row i of matrix Q^k that is an encryption of 1. Since the encryption scheme E has an additive homomorphism and we know already that $E^{-1}(Q^k_{i,j}) \in \{0,1\}$, the proof is $\mathsf{NIZKP}\left\{E^{-1}(\prod_{j=1}^{t} Q^k_{i,j}) \in \{1\}\right\}$.
4. The voter proves that the pointers are well-formed; that is, for all $k \in \{1, \ldots, t\}$ we have $p^k = F^{-1}(C_k) + F^{-1}(\overrightarrow{a}^k_1)$. Formally, the voter gives $\mathsf{NIZKP}\left\{F^{-1}(C_k \cdot \overrightarrow{a}^k_1) \in \left\{p^k\right\}\right\}$.
5. The voter proves that the cyclic shift vectors are well-formed; that is, for all $k \in \{1, \ldots, t\}$ and all $i \in \{0, \ldots, t\}$, if $\overrightarrow{a}^k_i = F(j)$ then $Q^k_{i,j} = E(1)$. Formally, the proof is $\mathsf{NIZKP}\left\{F^{-1}(\overrightarrow{a}^k_i) \in \{j\}\right\} \bigvee \mathsf{NIZKP}\left\{E^{-1}(Q^k_{i,j}) \in \{0\}\right\}$.
6. The voter proves that the update vectors are well-formed; that is, show that for all $k \in \{1, \ldots, t\}$ and all $i \in \{0, \ldots, t\}$, we have $\overrightarrow{u}^k_i = F(-1)$ if $F^{-1}(C_i) < F^{-1}(C_k)$ and $\overrightarrow{u}^k_i = F(0)$ otherwise. Formally, the voter gives

$$\Big(\bigvee_{\lambda \in \{0,\ldots,t-1\}} \big(\mathsf{NIZKP}\left\{F^{-1}(C_k) \in \{\lambda\}\right\} \bigwedge \mathsf{NIZKP}\left\{F^{-1}(C_i) \in \{0, \ldots, \lambda - 1\}\right\}\big) \bigwedge \mathsf{NIZKP}\left\{F^{-1}(\overrightarrow{u}^k_i) \in \{-1\}\right\}\Big) \bigvee \mathsf{NIZKP}\left\{F^{-1}(\overrightarrow{u}^k_i) \in \{0\}\right\}.$$

Acknowledgments

The authors would like to thank Kobbi Nissim and the anonymous reviewers for their comments.

References

1. R. Aditya, C. Boyd, E. Dawson, and K. Viswanathan. Secure e-voting for preferential elections. In R. Traunmller, editor, *Proceedings of Electronic Government 2003*, volume 2739 of *LNCS*, pages 246–249, 2003.
2. O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In N. Shavit, editor, *Proceedings of the ACM Symposium on the Principles of Distributed Systems 2001*, pages 274–283, 2001.
3. J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1987.
4. J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas in Cryptography 1994*, pages 120–128, May 1994.

5. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 544–553, 1994.
6. J. Benaloh and M. Yung. Distributing the power of a government to enhance to privacy of voters. In *Proceedings of the 5th Symposium on Principles of Distributed Computing*, pages 52–62, 1986.
7. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
8. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, pages 372–382, 1985.
9. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Proceedings of Crypto 1994*, volume 893 of *LNCS*, pages 174–187, 1994.
10. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 72–83, 1996.
11. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, Sep 1997.
12. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136, 2001.
13. T. ElGamal. A public key cryptosystem and a signature scheme based on disc rete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko, editor, *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194, 1986.
15. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In Y. Frankel, editor, *Proceedings of Financial Cryptography 2000*, volume 1962 of *LNCS*, pages 90–104, 2000.
16. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Proceedings of Financial Cryptography 2002*, pages 16–30, 2002.
17. S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th ACM Symposium on Theory of computing*, pages 365–377, 1982.
18. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 539–556, 2000.
19. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In D. Boneh, editor, *Proceedings of USENIX Security Symposium 2002*, pages 339–353, 2002.
20. J. Katz, S. Myers, and R. Ostrovsky. Cryptographic counters and applications to electronic voting. In B. Pfitzmann, editor, *Proceedings of Eurocrypt 2001*, volume 2045, pages 78–92, 2001.
21. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Symposium on Computer and Communications Security*, pages 59–66, 1998.
22. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–238, 1999.

23. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 387–398, 1996.
24. M. Rabin. Transaction protection by beacons. *Journal of Computer and System Science*, 27(2):256–267, 1983.
25. K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In L. C. Guillou and J.-J. Quisquater, editors, *Proceedings of Eurocrypt 1995*, volume 921 of *LNCS*, pages 393–403, 1995.
26. A. D. Santis, G. D. Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proceedings of the IEEE Symposium on Foundations of Computer Science 1994*, pages 454–465, 1994.

A Privacy Game 1 (Multiple Counters)

Privacy Game 1 (for z counters)

Setup: same as Privacy Game 0, except that $\mathcal{A}$ outputs two sets of z initial values $\overrightarrow{V}^*, \overrightarrow{V}' \in \mathbb{Z}_M^z$ and their corresponding sets of update values $\overrightarrow{U}^* \in \mathbb{Z}_M^{zk}$ and $\overrightarrow{U}' \in \mathbb{Z}_M^{zk}$ for the events.

Challenge: $\mathcal{C}$ flips a random bit b. $\mathcal{A}$ constructs the challenge set of private counters C_b from $\mathcal{SK}_E, \mathcal{SK}_F$, and $\mathcal{A}$'s parameters in the following way — If $b = 0$, $\mathcal{C}$ constructs z private counters using initial values in $\overrightarrow{V}^*$ and the update values in $\overrightarrow{U}^*$; if $b = 1$, $\mathcal{C}$ constructs z private counters with initial values in $\overrightarrow{V}'$ and update values in $\overrightarrow{U}'$.

Queries: $\mathcal{A}$ can request invocations to the function Apply from $\mathcal{C}$.

Output: $\mathcal{A}$ outputs its guess g for the bit b.

We say that $\mathcal{A}$ wins privacy game 1 if $\mathcal{A}$ guesses bit b correctly.

Definition 2. *A counter scheme is private according to game 1 if all polynomial (in security parameter t) time algorithms win game 1 only with negligible advantage* $\mathtt{Adv}(t) = |\Pr[g = b] - 1/2|$.

Proposition 2. *If a counter scheme is private according to Game 1, then that same counter scheme is private according to Game 0.*

The proof follows from a standard hybrid argument.