Constant-Round Private Function Evaluation with Linear Complexity

Jonathan Katz[1,⋆] and Lior Malka[2,⋆⋆]

[1] Dept. of Computer Science,
University of Maryland
jkatz@cs.umd.edu
[2] Intel
lior34@gmail.com

Abstract. We consider the problem of *private function evaluation* (PFE) in the two-party setting. Here, informally, one party holds an input x while the other holds a (circuit describing a) function f; the goal is for one (or both) of the parties to learn $f(x)$ while revealing nothing more to either party. In contrast to the usual setting of secure computation, where the function being computed is known to both parties, PFE is useful in settings where the function (i.e., algorithm) itself must remain secret, e.g., because it is proprietary or classified.

It is known that PFE can be reduced to standard secure computation by having the parties evaluate a *universal circuit*, and this is the approach taken in most prior work. Using a universal circuit, however, introduces additional overhead and results in a more complex implementation. We show here a completely new technique for PFE that avoids universal circuits, and results in constant-round protocols with communication/computational complexity *linear* in the size of the circuit computing f. This gives the first constant-round protocol for PFE with linear complexity (without using fully homomorphic encryption), even restricted to semi-honest adversaries.

1 Introduction

In the setting of two-party *private function evaluation* (PFE), a party P_1 holds an input x while another party P_2 holds a (circuit C_f describing a) function f; the goal is for one (or both) of the parties to learn the result $f(x)$ while not revealing to either party any information beyond this. (The parties do agree in advance on the *size* of the circuit being computed, as well as the input/output length. See Section 2.1 for further discussion.) PFE is useful when the function being computed must remain private, say because the function is classified, because revealing the function would lead to security vulnerabilities, or because the *implementation* of the function (e.g., the circuit C_f itself) is proprietary even if the function f is known [34, 6, 8, 9, 11–13, 19, 5, 33, 31, 3].

⋆ Work supported by DARPA and NSF award #1111599.
⋆⋆ Work done while at the University of Maryland.

PFE stands in contrast to the standard setting of secure two-party computation [37, 14], where the parties hold inputs x and y, respectively, and wish to compute the result $f(x, y)$ for some *mutually known* function f using an agreed-upon circuit C_f for computing f. On the other hand, it is well known that the problem of PFE can be reduced to the problem of secure computation using *universal circuits*. In more detail, let U_n be some (fixed) universal circuit such that $U_n(x, C) = C(x)$ for every circuit C having at most n gates. (We implicitly assume here some fixed representation for circuits.) Then if $\mathcal{C}_n$ is the class of circuits having at most n gates, PFE for this class is solved by having the parties run a (standard) secure computation of U_n.

There are, however, drawbacks to using universal circuits to implement PFE. First is the resulting complexity: although PFE using universal circuits has been implemented [35], it is fair to say that it is more challenging, tedious, and error-prone to write code involving universal circuits than it is to implement secure computation "directly" using Yao's garbled circuit approach (as done, e.g., in [27, 26, 32, 16, 17]). Using universal circuits also impacts efficiency. Valiant [36] showed a construction of a universal circuit achieving (optimal) $|U_n| = O(n \log n)$; the construction is complex, however, and the constant terms (as well as the low-order terms) are significant. Kolesnikov and Schneider [23, 35] gave a simpler construction of universal circuits: they obtain the worse asymptotic bound $|U_n| = O(n \log^2 n)$, but their techniques are claimed to yield smaller universal circuits than Valiant's construction for "reasonable" values of n. (The exact improvement depends also on the number of inputs and outputs. We refer the reader to their work for a detailed comparison.) Even so, as secure two-party computation is used for ever-larger circuits (secure computation of circuits with up to 1 billion gates has been reported [17]), the overhead introduced by universal circuits becomes prohibitive. Indeed, the implementation of PFE by Kolesnikov and Schneider [23, 35] can handle circuits of only a few thousand gates [31].

Another approach to PFE is given by Abadi and Feigenbaum [1], who show a PFE protocol with complexity $O(n)$ but using $O(d)$ rounds, where d is (an upper bound on) the depth of the circuit being computed.

1.1 Contributions of Our Work

We show the first *constant-round* PFE protocols with *linear* complexity, without relying on fully homomorphic public-key encryption.[1] We begin by showing a protocol in the semi-honest setting; this illustrates our core techniques and represents what we consider to be our main contribution. (Semi-honest security was the focus of all prior work on PFE [34, 6, 8, 9, 11–13, 19, 5, 33, 31, 3].) Zero-knowledge proofs can be used in the standard way [15] to obtain security against malicious parties, still in constant rounds and with linear complexity; however, the resulting protocol is unlikely in practice to out-perform secure computation

[1] It is easy to construct constant-round, linear-complexity PFE from fully homomorphic encryption. But it is of theoretical interest to reduce the assumptions used, and of practical importance to avoid the overhead of fully homomorphic encryption.

of universal circuits using efficient protocols for the malicious setting (e.g., [24]). We sketch a more efficient construction for achieving security against a malicious P_1.

Our protocols rely on (singly) homomorphic public-key encryption as well as a symmetric-key encryption scheme secure against *linear* related-key attacks; see Definition 2. The former can be instantiated using various standard cryptosystems (e.g., [10, 30]); the latter can be instantiated in the random oracle model, or in a provable sense [2] based on the decisional Diffie-Hellman assumption.

In addition to the theoretical improvement, our approach should yield better performance in practice for PFE of large circuits and/or in certain settings. Specifically, although our protocol uses $O(n)$ public-key operations — in contrast to universal-circuit-based approaches that use $O(n \log n)$ or $O(n \log^2 n)$ symmetric-key operations[2] — our protocol has linear communication complexity, making it advantageous when network communication is expensive. Moreover, there are several ways our protocol can be improved (e.g., using elliptic-curve cryptography along with fast algorithms for performing multiple fixed-base exponentiations) to reduce its computational cost.

1.2 Overview of Our Techniques

Our main technical contribution, as noted above, is our idea for achieving PFE with linear complexity in the semi-honest setting; we describe this here. Our description is fairly detailed and we will refer to it in the formal description of our protocol later; it should also be possible to skim this section so as to obtain the main ideas. Our approach adapts Yao's garbled-circuit technique. At a very high level, our idea is to have P_1 generate a sequence of gates; P_2 then connects these gates together, using (singly) homomorphic encryption, in a manner that is oblivious to P_1, while still enabling P_1 to prepare a garbled circuit corresponding to the circuit C_f held by P_2. This idea of having one party connect gates of the circuit together is vaguely reminiscent of the "soldering" approach taken in [29]; our setting, however, is different than theirs (in [29] it was required that both parties know the circuit being computed), as is our implementation of the "soldering" step.

Say $x \in \{0,1\}^\ell$, and assume that f outputs a single bit and that C_f is known to contain exactly n NAND gates. (Neither of these assumptions is necessary, but we avoid complications for now.) It will be useful to distinguish between *outgoing wires* and *ingoing wires* of a circuit. Outgoing wires include the ℓ input wires of the circuit, along with the wire that exits each gate of the circuit; thus, in a circuit with ℓ inputs and n gates there are exactly $\ell + n$ outgoing wires. The ingoing wires are exactly the input wires to each gate of the circuit; thus, in a circuit with n two-input gates there are exactly $2n$ ingoing wires. A circuit is defined by specifying the output wires, and by giving a correspondence between

[2] This does not account for any oblivious transfers performed in the universal-circuit-based approaches. However the number of oblivious transfers scales linearly in the input length, not the circuit size.

outgoing wires and ingoing wires; e.g., specifying that outgoing wire i (which may be an input wire or a wire exiting some gate) connects to ingoing wires j, k, and ℓ. We stress that even though we speak of each internal gate as having only a single outgoing wire, we handle arbitrary fan-out since a single outgoing wire can be connected to several ingoing wires.

In our description below, we assume for concreteness that P_2 learns the output $f(x)$. However, it is trivial to modify our protocol (with no additional cost) so that only P_1 learns the output. See the remark at the end of this section.

The protocol begins by having P_1 generate and send a public key pk for a (singly) homomorphic encryption scheme Enc. Similar to Yao's garbled-circuit technique, P_1 then chooses $\ell + n$ pairs of random *keys* that will be assigned to each of the outgoing wires. Let s_i^b denote the key corresponding to bit b on wire i. Then P_1 sends

$$\left[\mathsf{Enc}_{pk}(s_1^0), \mathsf{Enc}_{pk}(s_1^1)\right], \ldots, \left[\mathsf{Enc}_{pk}(s_{\ell+n}^0), \mathsf{Enc}_{pk}(s_{\ell+n}^1)\right]$$

to P_2. (It will become clear from what follows that P_1 need not send the final encrypted pair $\left[\mathsf{Enc}_{pk}(s_{\ell+n}^0), \mathsf{Enc}_{pk}(s_{\ell+n}^1)\right]$. We include it above for clarity.)

P_2, in turn, *obliviously* defines keys for each of the $2n$ ingoing wires. P_2 sorts the gates of C_f topologically, so that if the outgoing wire from some gate i connects to an ingoing wire of some gate j then $i < j$. This defines a natural enumeration of the outgoing wires in the circuit: outgoing wires numbered from 1 to ℓ correspond to the input wires of the circuit, and outgoing wire $\ell + i$ (for $i \in \{1, \ldots, n\}$) corresponds to the wire exiting gate i. The output wire of the circuit corresponds to outgoing wire $\ell + n$. (Recall that here we assume f is boolean; in Section 3.1 we relax this.)

For each ingoing wire of the circuit, P_2 does as follows. Say the ingoing wire of some gate i is connected to outgoing wire j. Then P_2 chooses random a_i, b_i and defines the (encrypted) keys for this ingoing wire to be

$$\left[\mathsf{Enc}_{pk}(a_i \cdot s_j^0 + b_i),\ \mathsf{Enc}_{pk}(a_i \cdot s_j^1 + b_i)\right],$$

where the above is computed using the homomorphic properties of the encryption scheme. (In the above, the ciphertexts are re-randomized in the usual way.) Two observations are in order: first, the (unencrypted) keys $(r^0, r^1) \overset{\text{def}}{=} \left(a_i \cdot s_j^0 + b_i,\ a_i \cdot s_j^1 + b_i\right)$ are random and independent of j. Second, given s_j^b it is possible for P_2 to compute r^b (using a_i, b_i); without s_j^{1-b}, however, P_2 learns no information about r^{1-b}. (Recall we are in the semi-honest setting, so a_i, b_i are chosen at random.)

Expanding upon the above, say gate i of the circuit has its left ingoing wire connected to outgoing wire j and right ingoing wire connected to outgoing wire k. (As always, the outgoing wire from this gate is numbered $\ell + i$.) Then P_2 defines the encrypted "garbled gate"

$$\mathsf{encGG}_i = \begin{pmatrix} \left[\mathsf{Enc}_{pk}(a_i \cdot s_j^0 + b_i),\ \mathsf{Enc}_{pk}(a_i \cdot s_j^1 + b_i)\right] \\ \left[\mathsf{Enc}_{pk}(a_i' \cdot s_k^0 + b_i'),\ \mathsf{Enc}_{pk}(a_i' \cdot s_k^1 + b_i')\right] \\ \left[\mathsf{Enc}_{pk}(s_{\ell+i}^0),\ \mathsf{Enc}_{pk}(s_{\ell+i}^1)\right] \end{pmatrix},$$

where a_i, b_i, a'_i, b'_i are chosen uniformly at random. Finally, P_2 sends

$$\mathsf{encGG}_1, \ldots, \mathsf{encGG}_n$$

to P_1. (In fact P_2 need not transmit the final pair $\left[\mathsf{Enc}_{pk}(s^0_{\ell+i}),\ \mathsf{Enc}_{pk}(s^1_{\ell+i})\right]$ of each encrypted garbled gate, since P_1 knows it. We include it above for clarity.)

Upon receiving this message, P_1 decrypts each encGG to obtain, for each gate i, the three pairs of keys $\left([L^0_i, L^1_i], [R^0_i, R^1_i], [s^0_{\ell+i}, s^1_{\ell+i}]\right)$. It then prepares a garbled version GG_i of this gate in the usual way: namely, it computes the four ciphertexts

$$C'_{b,c} \leftarrow \mathsf{sEnc}_{L^b_i}\left(\mathsf{sEnc}_{R^c_i}\left(s^{\mathrm{NAND}(b,c)}_{\ell+i}\right)\right), \quad b, c \in \{0,1\}$$

(where sEnc denotes a symmetric-key encryption scheme), and sets GG_i to be the four ciphertexts $\left(C'_{0,0}, \ldots, C'_{1,1}\right)$ in random permuted order. P_1 then sends $\mathsf{GG}_1, \ldots, \mathsf{GG}_n$ to P_2. In addition, P_1 sends the appropriate input-wire keys $s^{x_1}_1$, $\ldots$, $s^{x_\ell}_\ell$, as well as both output-wire keys $\left(s^0_{\ell+n}, s^1_{\ell+n}\right)$.

P_2 now has enough information to compute the result, using a procedure analogous (but not identical) to what is done in a standard application of Yao's garbled-circuit methodology. P_2 begins knowing a key s_i for each outgoing wire $i \in \{1, \ldots, \ell\}$. (Recall these are the input wires of the circuit that correspond to P_1's input.) Inductively, P_2 can compute a key for every outgoing wire as follows: Consider the $(\ell + i)$th outgoing wire exiting from gate i, where the left ingoing wire to this gate is connected to outgoing wire $j < i$ and the right ingoing wire to this gate is connected to outgoing wire $k < i$. Assume P_2 has already determined keys s_j, s_k for outgoing wires j, k, respectively. P_2 computes keys $L_i = a_i s_j + b_i$ and $R_i = a'_i s_k + b'_i$ for the left and right *ingoing* wires to gate i. Then P_2 tries to decrypt each of the four ciphertexts in GG_i. With overwhelming probability, only one of these decryptions will be successful; the result of this successful decryption defines the key $s_{\ell+i}$ for outgoing wire $\ell + i$. Once P_2 has determined key $s_{\ell+n}$, it can check whether this corresponds to an output of '0' or '1' using the ordered pair $\left(s^0_{\ell+n}, s^1_{\ell+n}\right)$ sent by P_1.

Further details, intuition for security of the above, proofs of security, and extensions to handle malicious behavior of P_1 are described in the sections that follow. A more efficient variant of the above protocol is described in Section 3.2.

Remark 1: It is trivial to modify the above protocol, at no additional cost, so that only P_1 learns the output (and P_2 learns nothing): first, change round 3 so that P_1 does not send the output-wire keys $\left(s^0_{\ell+n}, s^1_{\ell+n}\right)$. Then when P_2 learns the final key $s_{\ell+n}$ it simply sends this key back to P_1, who can then check whether it is equal to $s^0_{\ell+n}$ or $s^1_{\ell+n}$.

1.3 Other Related Work

Several works have explored weaker variants of PFE. Paus et al. [31] consider *semi-private* function evaluation where the circuit *topology* (i.e., the connections between gates) is assumed to be known to both parties, but the boolean function computed by each gate can be hidden. Here we treat the more difficult case where *everything* about the circuit (except an upper bound on its size and the number of inputs/outputs) is hidden. Another direction has been to consider PFE for limited classes of functions: e.g., functions defined by low-depth circuits [34, 4], branching programs [19, 3], or polynomials [9, 28]. Here we handle functions defined by arbitrary (polynomial-size) circuits.

2 Definitions

Let k be the security parameter. A *distribution ensemble* $X = \{X(1^k, a)\}_{k\in\mathbb{N},\, a\in\mathcal{D}}$ is an infinite sequence of random variables indexed by $k \in \mathbb{N}$ and $a \in \mathcal{D}$, for $\mathcal{D}$ some specified set. The two ensembles $X = \{X(1^k, a)\}_{k\in\mathbb{N},\, a\in\mathcal{D}}$ and $Y = \{Y(1^k, a)\}_{k\in\mathbb{N},\, a\in\mathcal{D}}$ are *computationally indistinguishable*, denoted $X \overset{c}{\equiv} Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function $\mu(\cdot)$ such that for every k and every $a \in \mathcal{D}$

$$\Big|\Pr[D(X(1^k, a)) = 1] - \Pr[D(Y(1^k, a)) = 1]\Big| \le \mu(k).$$

2.1 Private Function Evaluation

Our definitions of security are standard, but we include them here for completeness. For simplicity, we treat the case where P_1 holds some value $x \in \{0,1\}^\ell$ as input while P_2 holds a circuit C_f computing some deterministic function f; the goal of the protocol is for P_2 to learn $f(x)$. The definitions we provide here, as well as our protocols, extend easily to handle, e.g., additional input provided by P_2 (this can simply be incorporated into the circuit C_f), randomized functions f, or the case where P_1 receives output (see Remark 1 at the end of Section 1.2).

The problem of PFE is meaningless in practice if P_2 learns the output and f (resp., C_f) is allowed to be completely arbitrary: in that case P_2 could take $f(x) = x$ and learn P_1's entire input! It is thus reasonable to impose some restrictions on C_f. The most general formulation to assume that both parties fix some class $\mathcal{C}$ of circuits, and require that $C_f \in \mathcal{C}$; in that case we refer to the problem as $\mathcal{C}$-PFE. This encompasses both the case when P_1 knows some partial information about f (as in [31]), as well as the case where C_f is restricted in some way (e.g., to have low depth). In this work, we assume only that P_1 knows the input length ℓ, and upper bounds on the output length m and the number of gates n (i.e., $\mathcal{C}$ contains only circuits satisfying those constraints). Note that if $m \ll \ell$ then meaningful privacy of P_1's input is maintained regardless of what circuit $C_f \in \mathcal{C}$ is used by P_2.

There are two ways one could incorporate a security parameter k into the definition of the problem. The usual way, which we find less natural in our setting, is to allow the sizes of the inputs to grow and to set the security parameter equal to the input size(s). We prefer instead to treat the input domains (namely, $\{0,1\}^\ell$ and some class of circuits $\mathcal{C}$) as fixed, and to treat k as an additional input.

A two-party protocol for $\mathcal{C}$-PFE is a protocol running in polynomial time and satisfying the following correctness requirement: if party P_1, holding input 1^k and x, and party P_2, holding input 1^k and $C_f \in \mathcal{C}$, run the protocol honestly, then (except with probability negligible in k) the output of P_2 is $C_f(x)$.

Security in the semi-honest case. In the semi-honest case we assume both parties follow the protocol honestly but may each try to learn some additional information from their (respective) view. Fix $\mathcal{C}$ and let Π be a protocol for $\mathcal{C}$-PFE. The *view* of the *i*th party during an execution of Π when the parties begin holding inputs x and C_f, respectively, and security parameter 1^k is denoted by $\text{VIEW}_i^{\Pi}(1^k, x, C_f)$. The view of P_i contains P_i's input and random tape, along with the sequence of messages received from the other party P_{3-i}.

When f is deterministic it suffices to consider the views of the parties in isolation, rather than their joint distribution [14, Sect. 7.2.2.1]. We thus have:

Definition 1. *Protocol Π is a* secure $\mathcal{C}$-PFE protocol for semi-honest adversaries *if there exist probabilistic polynomial-time simulators $\mathcal{S}_1, \mathcal{S}_2$ such that*

$$\left\{\mathcal{S}_1\left(1^k, x\right)\right\}_{k\in\mathbb{N},\, x\in\{0,1\}^\ell,\, C_f\in\mathcal{C}} \overset{c}{\equiv} \left\{\text{VIEW}_1^{\Pi}\left(1^k, x, C_f\right)\right\}_{k\in\mathbb{N},\, x\in\{0,1\}^\ell,\, C_f\in\mathcal{C}}$$

$$\left\{\mathcal{S}_2\left(1^k, C_f, C_f(x)\right)\right\}_{k\in\mathbb{N},\, x\in\{0,1\}^\ell,\, C_f\in\mathcal{C}} \overset{c}{\equiv} \left\{\text{VIEW}_2^{\Pi}\left(1^k, x, C_f\right)\right\}_{k\in\mathbb{N},\, x\in\{0,1\}^\ell,\, C_f\in\mathcal{C}} .$$

Security against malicious behavior. We refer to the full version of this paper [21] for a definition of security against malicious adversaries within the usual real/ideal framework [14].

2.2 Tools

We use a (singly) homomorphic public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The actual property we need is the ability to evaluate a *pairwise-independent function* on the plaintext space. If the plaintext space is a group $\mathbb{G}$ of prime order p, written additively, this can be achieved by mapping $a \in \mathbb{Z}_p$, $b \in \mathbb{G}$, and $\mathsf{Enc}_{pk}(m)$ to a (random) encryption of $\mathsf{Enc}_{pk}(am + b)$. Thus, e.g., standard El Gamal encryption [10] can be used (though $\mathbb{G}$ in that case is usually written multiplicatively). In fact, the plaintext space is not required to have prime order, as we only require "almost" pairwise-independence. In particular, Paillier encryption [30] could also be used.

We also use a symmetric-key encryption scheme $(\mathsf{sEnc}, \mathsf{sDec})$ whose key space is viewed as a group $\mathbb{G}(k)$ of order $p = p(k)$ that is, for simplicity, the same as the plaintext space of the public-key encryption scheme being used. (In practice, this can be achieved for any desired $\mathbb{G}$ by implementing encryption with key $g \in \mathbb{G}$ using AES with key SHA-1(g), truncated to 128 bits.) We impose the same requirements on $(\mathsf{sEnc}, \mathsf{sDec})$ as in [25]: namely, that it have *elusive* and *efficiently verifiable* range. (These properties are easily satisfied.) In addition, we require $(\mathsf{sEnc}, \mathsf{sDec})$ to satisfy a weak form of *related-key security* where, roughly, encryption remains secure even when performed using linearly related keys (where the linear relations are chosen at random). That is:

Definition 2. *Encryption scheme* $(\mathsf{sEnc}, \mathsf{sDec})$ *is* secure against linear related-key attacks *if the following is negligibly close (in k) to* $1/2$ *for all polynomials d and all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l} s \leftarrow \mathbb{G}(k); c \leftarrow \{0,1\}; \\ a_1, \ldots a_d \leftarrow \mathbb{Z}_{p(k)} \\ b_1, \ldots, b_d \leftarrow \mathbb{G}(k) \end{array} : \mathcal{A}^{\mathsf{sEnc}^c_{a_1 s + b_1}(\cdot,\cdot), \ldots, \mathsf{sEnc}^c_{a_d s + b_d}(\cdot,\cdot)}(a_1, b_1, \ldots, a_d, b_d) = c\right],$$

where $\mathsf{sEnc}^c_s(m_0, m_1) \overset{\text{def}}{=} \mathsf{sEnc}_s(m_c)$*.*

We remark that a weaker definition (where $\mathcal{A}$ queries each $\mathsf{sEnc}^c_{a_i s + b_i}(\cdot,\cdot)$ only on two inputs, chosen nonadaptively) suffices for our proof. It is easy to construct an encryption scheme satisfying the above definition using a (non-programmable) random oracle, and it would be surprising if standard encryption schemes based on AES could be shown not to satisfy the above definition. Moreover, recent work of Applebaum et al. [2] can be used to construct a scheme satisfying the above definition in a provable sense, based on the decisional Diffie-Hellman assumption.

3 A $\mathcal{C}$-PFE Protocol for Semi-honest Adversaries

3.1 Description of the Protocol

We now formally define our $\mathcal{C}$-PFE protocol for semi-honest adversaries. In our description here, we assume the reader is familiar with the protocol overview provided in Section 1.2.

We assume that all circuits in $\mathcal{C}$ are composed solely of NAND gates. This is for simplicity only, and our protocol can be easily modified to handle circuits over an arbitrary basis of 2-to-1 gates with only a small impact on the efficiency. Let n be an upper bound on the size of any circuit in $\mathcal{C}$, and let m be an upper bound on the number of outputs. By adjusting n appropriately, we may assume that every circuit in $\mathcal{C}$ has exactly m outputs (P_2 can always add "dummy" outputs that are fixed to some constant); that the output wires of the circuit do not connect to any other gates (this can be achieved by adding at most m gates to the circuit); and that every circuit in $\mathcal{C}$ contains exactly n gates (P_2 can add "dummy" gates whose output wires are connected to nothing). We make all these assumptions in what follows. We also assume that P_2 learns the output; however, it is trivial to modify the protocol so that P_1 learns the output; see Remark 1 in Section 1.2.

Recall from Section 1.2 that we distinguish between *outgoing wires* and *ingoing wires* of C_f. (Recall also that although each gate has only a single outgoing wire, we handle circuits with arbitrary fan-out since a single outgoing wire can be connected to several ingoing wires.) As in Section 1.2, party P_2 sorts the gates of C_f topologically and this defines an enumeration of the $N \overset{\text{def}}{=} \ell + n$ outgoing wires. The outgoing wires numbered from 1 to ℓ correspond to the ℓ input wires of the circuit, and outgoing wire $\ell + i$ (for $i \in \{1, \ldots, n\}$) corresponds to the output wire from gate i. The output wires of the circuit correspond to the m outgoing wires $N-m+1,\ldots,N$.

We first define an algorithm encYao that prepares garbled gates as in Yao's protocol: encYao takes as input three pairs of keys and outputs four ciphertexts, and is defined as

$$\mathsf{encYao}\left([L^0,L^1],[R^0,R^1],[s^0,s^1]\right) \overset{\text{def}}{=} \left\{\mathsf{sEnc}_{L^b}\left(\mathsf{sEnc}_{R^c}\left(s^{\text{NAND}(b,c)}\right)\right)\right\}_{b,c\in\{0,1\}},$$

where the four ciphertexts are in random permuted order. We analogously define an algorithm decYao that takes as input two keys (for each of two ingoing wires) and a garbled gate, and outputs a key for the outgoing wire; this algorithm, given keys L,R and four ciphertexts $\{C'_0,C'_1,C'_2,C'_3\}$, computes $\mathsf{sDec}_L(\mathsf{sDec}_R(C'_i))$ for all i and outputs the unique non-$\perp$ value that is obtained. (If more than one non-$\perp$ value results, this algorithm outputs $\perp$.)

Our protocol is described in Figure 1. Correctness holds with all but negligible probability, via an argument similar to the one in [25].

In our description of the protocol we aimed for clarity rather than efficiency, and several improvements are possible. For one, P_2 need not include $\left[\mathsf{Enc}_{pk}(s^0_{\ell+i}),\ \mathsf{Enc}_{pk}(s^1_{\ell+i})\right]$ as part of encGG_i since P_1 already knows these values. Furthermore, P_1 need not send

$$\left[\mathsf{Enc}_{pk}(s^0_{N-m+1}),\mathsf{Enc}_{pk}(s^1_{N-m+1})\right],\ldots,\left[\mathsf{Enc}_{pk}(s^0_N),\mathsf{Enc}_{pk}(s^1_N)\right]$$

in round 1 (since these outgoing wires do not connect to any ingoing wires). Moreover, P_1 can set $s^0_{N-m+1}=\cdots=s^0_N=0$ and $s^1_{N-m+1}=\cdots=s^1_N=1$ (and then there is no need to send the output-wires message in the third round); that is, for gates whose outgoing wires are the output of the circuit, P_1 can encrypt the wire value itself rather than encrypting a key that encodes the wire value.

Security against a semi-honest P_1 is easy to see. In fact, security in that case holds in a *statistical* sense. Indeed, with all but negligible probability it holds that $s^0_i \neq s^1_i$ for all $i\in\{1,\ldots,N\}$. Assuming this to be the case, the top two rows of each encGG_i sent by P_2 to P_1 in round 2 consist only of (random) encryptions of the four independent, uniform values

$$a_i\cdot s^0_j+b_i,\quad a_i\cdot s^1_j+b_i,\quad a'_i\cdot s^0_k+b'_i,\quad a'_i\cdot s^1_k+b'_i.$$

In particular, these values are independent of the interconnections between gates of C_f, and thus the view of P_1 is independent of the circuit held by P_2.

Security against a semi-honest P_2 holds *computationally*, assuming semantic security of the homomorphic encryption scheme and security against linear related-key attacks for the symmetric-key encryption scheme. Roughly, the initial encryptions sent to P_2 in round 1 do not reveal anything about the values s^0_i,s^1_i that P_1 assigns to each outgoing wire in the circuit. Thus, the information sent to P_2 in round 3 is essentially equivalent to the information sent to P_2 in a standard application of Yao's garbled-circuit methodology, with the only difference being that here ingoing wires and outgoing wires have different keys, and

Inputs: The security parameter is k. The input of P_1 is a value $x \in \{0,1\}^\ell$, and the input of P_2 is a circuit C_f with ℓ, n, m as described in the text.

Round 1 P_1 computes $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ and sends pk to P_2. In addition, P_1 chooses $N = \ell + n$ pairs of random keys s_i^0, s_i^1 for $i \in \{1, \ldots, N\}$. It then sends to P_2 the ciphertexts

$$\left[\mathsf{Enc}_{pk}(s_1^0), \mathsf{Enc}_{pk}(s_1^1)\right], \ldots, \left[\mathsf{Enc}_{pk}(s_N^0), \mathsf{Enc}_{pk}(s_N^1)\right].$$

Round 2 For each gate $i \in \{1, \ldots, n\}$ of C_f, with left ingoing wire connected to outgoing wire j, right ingoing wire connected to outgoing wire k, and outgoing wire $\ell + i$, party P_2 chooses a_i, b_i, a_i', b_i' uniformly (from the appropriate domains) and computes

$$\mathsf{encGG}_i = \begin{pmatrix} \left[\mathsf{Enc}_{pk}(a_i \cdot s_j^0 + b_i),\ \mathsf{Enc}_{pk}(a_i \cdot s_j^1 + b_i)\right] \\ \left[\mathsf{Enc}_{pk}(a_i' \cdot s_k^0 + b_i'),\ \mathsf{Enc}_{pk}(a_i' \cdot s_k^1 + b_i')\right] \\ \left[\mathsf{Enc}_{pk}(s_{\ell+i}^0),\ \mathsf{Enc}_{pk}(s_{\ell+i}^1)\right] \end{pmatrix}$$

using the homomorphic properties of Enc. (In the above, each ciphertext is re-randomized.) Then P_2 sends $\mathsf{encGG}_1, \ldots, \mathsf{encGG}_n$ to P_1.

Round 3 For $i \in \{1, \ldots, n\}$, party P_1 decrypts encGG_i using sk to obtain the three pairs of keys $\mathsf{keys}_i \overset{\text{def}}{=} \left([L_i^0, L_i^1], [R_i^0, R_i^1], [s_{\ell+i}^0, s_{\ell+i}^1]\right)$. It then computes $\mathsf{GG}_i \leftarrow \mathsf{encYao}(\mathsf{keys}_i)$, and sends $\mathsf{GG}_1, \ldots, \mathsf{GG}_n$ to P_2. Finally, P_1 sends

$$\text{input-wires: } s_1^{x_1}, \ldots, s_\ell^{x_\ell}; \quad \text{output-wires: } \left(s_{N-m+1}^0, s_{N-m+1}^1\right), \ldots, \left(s_N^0, s_N^1\right).$$

Output determination Say P_1 sent input-wires: $s_1, \ldots, s_\ell$ to P_2 in the previous round. Then for all $i \in \{\ell+1, \ldots, \ell+n\}$, party P_2 does: If the left ingoing wire of gate i is connected to outgoing wire $j < i$ and the right ingoing wire of gate i is connected to outgoing wire $k < i$, then (1) compute $L_i = a_i s_j + b_i$ and $R_i = a_i' s_k + b_i'$, and then (2) set $s_i = \mathsf{decYao}(L_i, R_i, \mathsf{GG}_i)$.

Once P_2 has computed $s_1, \ldots, s_{\ell+n}$, it sets the jth output bit o_j (for $j \in \{N-m+1, \ldots, N\}$) to be the (unique) bit for which $s_j = s_j^{o_j}$.

Fig. 1. A $\mathcal{C}$-PFE protocol for semi-honest adversaries

P_2 must compute a key L_i on some ingoing wire by "translating" one of the keys s_j on the outgoing wire connected to that ingoing wire.

We have:

Theorem 1. *Assume the homomorphic encryption scheme is semantically secure, and the symmetric-key encryption scheme is secure against linear related-key attacks and has elusive and efficiently verifiable range. Then the protocol of Figure 1 is a secure $\mathcal{C}$-PFE protocol for semi-honest adversaries.*

Due to space limitations, a proof appears in the full version [21].

3.2 A More Efficient Variant

In this section we describe a more efficient variant of our protocol in which the wire labels are chosen in a coordinated fashion, as in [22]. Unfortunately, we are only able to prove security of the resulting protocol in the random oracle model; see further discussion at the end of this section.

We merely sketch the basic idea. Now, in round 1, P_1 chooses a global random shift r and $\ell + n$ outgoing-wire keys $\{s_i^0\}$; it then sets $s_i^1 = s_i^0 + r$ for all i. The first-round message from P_1 now contains pk and the $\ell + n$ ciphertexts $\mathsf{Enc}_{pk}(s_1^0), \ldots, \mathsf{Enc}_{pk}(s_{\ell+n}^0)$.

For each ingoing wire of the circuit, P_2 does as follows. Say this wire is connected to outgoing wire j. Then P_2 chooses random a and defines the (encrypted) 0-key for this ingoing wire to be (a re-randomization of) $\mathsf{Enc}_{pk}(s_j^0 + a)$, where this is computed using the homomorphic properties of the encryption scheme. Thus, if gate i of the circuit has its left ingoing wire connected to outgoing wire j and right ingoing wire connected to outgoing wire k, party P_2 defines the ith encrypted "garbled gate" via

$$\mathsf{encGG}_i = \begin{pmatrix} \mathsf{Enc}_{pk}(s_j^0 + a_i) \\ \mathsf{Enc}_{pk}(s_k^0 + a_i') \\ \mathsf{Enc}_{pk}(s_{\ell+i}^0) \end{pmatrix},$$

where a_i, a_i' are chosen uniformly at random. P_2 sends $\mathsf{encGG}_1, \ldots, \mathsf{encGG}_n$ to P_1.

Upon receiving this message, P_1 decrypts each encGG to obtain, for each gate i, the keys $\left(L_i^0, R_i^0, s_{\ell+i}^0\right)$. It defines $L_i^1 = L_i^0 + r$ and $R_i^1 = R_i^0 + r$, and then prepares a garbled version GG_i of this gate as in the previous sections. P_2 can then compute the result as usual. The entire protocol is roughly twice as efficient as the original.

As we have mentioned, however, we are only able to prove security of this modified protocol in the (non-programmable) random oracle model. Although it may appear possible to prove security in the standard model if the symmetric-key encryption scheme satisfies a strong enough definition of security, we were not able to isolate any suitable definition. In particular, correlation robustness [18] does not appear to suffice, since there is a circularity when, e.g., keys $s, s + r, s', s' + r$ are used to encrypt keys s'' and $s'' + r$. (Some combination of correlation robustness and circular security appears necessary.) The same issue seems to be present in the works of [22, 29] as well.

4 Security for Malicious Adversaries

As noted in the Introduction, we can apply zero-knowledge proofs in the standard way [15] to obtain a protocol with linear complexity (and constant round complexity) that is secure against malicious adversaries. However, the resulting protocol is unlikely in practice to out-perform secure computation of universal circuits using efficient protocols for the malicious setting (e.g., [24]). Here, we sketch a more efficient construction that achieves security against a malicious P_1 only. As in the previous section, our goal here is not to optimize the efficiency of the resulting protocol but rather to illustrate the main ideas.

We continue to assume that P_2 learns the output, however Remark 1 of Section 1.2 applies here as well and so the protocol is easily modified so that only P_1 learns the output.

4.1 Protocol Modifications

We introduce the following changes to the protocol described in Section 3.1:

Proof of well-formedness of pk. We require P_1 to prove that the public key pk it sends in round 1 was output by the specified key-generation algorithm Gen. (This step is not necessary if it is possible to efficiently verify whether a given pk could have been output by Gen, as is the case with, e.g., El Gamal encryption.) We remark further that it suffices for the proof to be honest-verifier zero knowledge (since we only require security against a semi-honest P_2), and we do not require it to be a proof of knowledge.

The complexity of this step is independent of n.

Validity of outgoing-wire keys. Let $\left[C_1^0, C_1^1\right], \ldots, \left[C_N^0, C_N^1\right]$ denote the ciphertexts sent by P_1 in round 1. (Recall that it is supposed to be the case that $C_i^b = \mathsf{Enc}_{pk}(s_i^b)$.) We now require P_1 to prove that (1) each C_i^b is a well-formed ciphertext with respect to the public key pk (once again, this step is unnecessary if it is possible to efficiently verify validity of ciphertexts, as is the case with El Gamal encryption), and (2) for each i, the ciphertexts C_i^0, C_i^1 are encryptions of *distinct* values. If the encryption scheme is additively homomorphic, and we let s_i^0 (resp., s_i^1) denote the plaintext corresponding to C_i^0 (resp., C_i^1), then P_2 can compute $\mathsf{Enc}_{pk}(s_i^0 - s_i^1)$ and the latter step is equivalent to proving that this is not an encryption of 0. Once again, it suffices for these proofs to be honest-verifier zero knowledge and they are not required to be proofs of knowledge.

The complexity of this step is linear in n since the statement being proved can be written as a conjunction of n statements, each of size independent of n.

Correctness of garbled-circuit construction. We require P_1 to prove correctness of the garbled gates it sends to P_2 in the final round. This amounts to proving, for each $i \in \{1, \ldots, n\}$, that GG_i was correctly constructed from encGG_i. As before, it suffices for these proofs to be honest-verifier zero knowledge and they are not required to be proofs of knowledge.

The complexity of this step is linear in n since the statement being proved is a conjunction of n statements, each of which has size independent of n. We also note that by using an appropriate homomorphic encryption scheme and symmetric-key encryption scheme, these proofs can be made (reasonably) efficient using the techniques of Jarecki and Shmatikov [20] (who show efficient proofs for exactly this purpose, assuming a common reference string, using a variant of the Camenisch-Shoup encryption scheme [7]).

Correctness of input-wire and output-wire keys. Finally, P_1 is required to prove that the input-wire and output-wire keys it sends in the final round are correct. Let $\left[C_1^0, C_1^1\right], \ldots, \left[C_N^0, C_N^1\right]$ denote the ciphertexts sent by P_1 in round 1 (recall it is supposed to be the case that $C_i^b = \mathsf{Enc}_{pk}(s_i^b)$), and let

$$\textsf{input-wires: } s_1, \ldots, s_\ell \quad \text{and} \quad \textsf{output-wires: } \left(s_{N-m+1}^0, s_{N-m+1}^1\right), \ldots, \left(s_N^0, s_N^1\right)$$

be the values sent by P_1 in the last round. Then P_1 must prove that: (1) that for each index $i \in \{1, \ldots, \ell\}$, one of the ciphertexts C_i^0, C_i^1 is an encryption of the plaintext s_i, and (2) that for each index $i \in \{N - m + 1, \ldots, N\}$, the ciphertext C_i^0 (resp., C_i^1) is an encryption of s_i^0 (resp., s_i^1). It suffices for each of these proofs to be honest-verifier zero knowledge; the first set of proofs (proving correctness of the input-wire keys) must be proofs of knowledge to allow for input extraction. (Alternately, if the proof of well-formedness of the public key is a proof of knowledge then proofs of knowledge are not needed here.)

The complexity of this step is linear in $\ell + m$.

We remark that most of the above proofs can be implemented efficiently for any homomorphic encryption scheme. The main exception is the proof of correctness of the garbled-circuit construction; however, as noted above, there exists at least one specific homomorphic encryption scheme for which this step can be done reasonably efficiently [20]. A proof of the following appears in [21].

Theorem 2. *Under the same assumptions as in Theorem 1, the protocol of Figure 1 with the modifications described in the previous section is a secure $\mathcal{C}$-PFE protocol for a malicious P_1.*

5 Conclusions and Future Work

We have shown the first constant-round protocol for PFE with complexity *linear* in the size of the circuit being computed (without relying on fully homomorphic encryption). Our results leave several interesting open questions:

– In addition to its theoretical importance, we believe our work is also of practical relevance: specifically, we expect that our approach to PFE will be both easier to implement and more efficient (for large circuits) than approaches relying on universal circuits. It remains to experimentally validate this claim.
– Our work leaves open the question of designing a fully secure protocol for PFE (i.e., PFE with security against a malicious P_1 and a malicious P_2) with linear complexity that would have better performance than what results from running a secure computation of universal circuits using efficient protocols for the malicious setting (e.g., [24]).
– It would also be interesting to further improve on the cryptographic assumptions needed for our results: e.g., to construct a protocol based on semantically secure symmetric-key encryption (without requiring related-key security), or to avoid the use of homomorphic public-key encryption.

The contents of this paper do not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. Journal of Cryptology 2(1), 1–12 (1990)
2. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. In: 2nd Symp. on Innovations in Computer Science, ICS (2011), `http://eprint.iacr.org/2010/544`
3. Barni, M., Failla, P., Kolesnikov, V., Lazzeretti, R., Sadeghi, A.-R., Schneider, T.: Secure Evaluation of Private Linear Branching Programs with Medical Applications. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 424–439. Springer, Heidelberg (2009)
4. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
5. Brickell, J., Porter, D.E., Shmatikov, V., Witchel, E.: Privacy-preserving remote diagnostics. In: 14th ACM Conf. on Computer and Communications Security (CCS), pp. 498–507. ACM Press (2007)
6. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-Round Secure Computation and Secure Autonomous Mobile Agents. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 512–523. Springer, Heidelberg (2000)
7. Camenisch, J., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
8. Canetti, R., Ishai, Y., Kumar, R., Reiter, M.K., Rubinfeld, R., Wright, R.N.: Selective private function evaluation with applications to private statistics. In: 20th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 293–304. ACM Press (2001)
9. Chang, Y.-C., Lu, C.-J.: Oblivious Polynomial Evaluation and Oblivious Neural Learning. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 369–384. Springer, Heidelberg (2001)
10. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory 31, 469–472 (1985)
11. Frikken, K., Atallah, M., Li, J.: Hidden access control policies with hidden credentials. In: Proc. ACM Workshop on Privacy in the Electronic Society (WPES), p. 27. ACM (2004)
12. Frikken, K., Attallah, M., Zhang, C.: Privacy-preserving credit checking. In: ACM Conf. on Electronic Commerce (EC), pp. 147–154. ACM (2005)
13. Frikken, K.B., Li, J., Atallah, M.J.: Trust negotiation with hidden credentials, hidden policies, and policy cycles. In: Network and Distributed System Security Symposium (NDSS), pp. 157–172. The Internet Society (2006)
14. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
15. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: 19th Annual ACM Symposium on Theory of Computing (STOC), pp. 218–229. ACM Press (1987)
16. Henecka, W., Kögl, S., Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: TASTY: Tool for automating secure two-party computations. In: 17th ACM Conf. on Computer and Communications Security (CCS), pp. 451–462. ACM Press (2010)
17. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: 20th USENIX Security Symposium (2011)

18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
19. Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
20. Jarecki, S., Shmatikov, V.: Efficient Two-Party Secure Computation on Committed Inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
21. Katz, J., Malka, L.: Constant-round private function evaluation with linear complexity, `http://eprint.iacr.org/2010/528`
22. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
23. Kolesnikov, V., Schneider, T.: A Practical Universal Circuit Construction and Secure Evaluation of Private Functions. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 83–97. Springer, Heidelberg (2008)
24. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
25. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology 22(2), 161–188 (2009)
26. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
27. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: Proc. 13th USENIX Security Symposium, pp. 287–302. USENIX Association (2004)
28. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM Journal on Computing 35(5), 1254–1281 (2006)
29. Nielsen, J.B., Orlandi, C.: LEGO for Two-Party Secure Computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
30. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
31. Paus, A., Sadeghi, A.-R., Schneider, T.: Practical Secure Evaluation of Semi-Private Functions. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 89–106. Springer, Heidelberg (2009)
32. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
33. Sadeghi, A.-R., Schneider, T.: Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 336–353. Springer, Heidelberg (2009)
34. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC^1. In: 40th Annual Symposium on Foundations of Computer Science (FOCS), pp. 554–567. IEEE (1999)

35. Schneider, T.: Practical secure function evaluation. Master's thesis, University Erlangen-Nürnberg (2008), `http://thomasckneider.de/FairplayPF`
36. Valiant, L.: Universal circuits. In: 8th Annual ACM Symposium on Theory of Computing (STOC), pp. 196–203. ACM Press (1976)
37. Yao, A.C.-C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (FOCS), pp. 162–167. IEEE (1986)