Secure circuit evaluation

A protocol based on hiding information from an oracle*

Martín Abadi
DEC Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301

Joan Feigenbaum
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Abstract: We present a simple protocol for two-player *secure circuit evaluation*. The protocol enables players C and D to cooperate in the computation of $f(x)$ while D conceals her data x from C and C conceals his circuit for f from D. The protocol is based on the technique of *hiding information from an oracle* [Abadi, Feigenbaum, Kilian, *J. Comput. System Sci.* 39(1):21–50, August, 1989].

1. Introduction

This paper describes a protocol for two-player *secure circuit evaluation*. We think of player C as a computer that runs a secret program for f and player D as another computer with some confidential data x. Suppose that D wishes to compute $f(x)$ but does not have an algorithm for f, and that C is willing to let D use his algorithm but does not want to reveal it. Alternatively, suppose that C has an algorithm, and possibly some confidential data as well, but needs D's confidential data in order to perform the computation. The protocol enables C and D to cooperate in the computation of $f(x)$ while D conceals her data x from C and C conceals his circuit for f from D; in fact, the circuit is hidden from D in an information-theoretic sense.

For example, suppose that C is a time-sharing computer and that D is one of C's users. Typically, when D wants to log in, she submits a name and a password, and C runs a program that checks whether they are correct. Of course, C does not have to divulge his program. Our protocol provides a way for D to log in without revealing her name or her password.

The protocol that we present is very simple. It allows player C to *hide* his circuit in the information-theoretic sense and player D to *encrypt* her inputs under a complexity-theoretic assumption. Our protocol uses the technique of *hiding information from an oracle* that was defined in our joint work with Kilian [AFK]. Specifically, we show how a function

*An extended abstract of this paper appeared in the Proceedings of the Fifth Annual Symposium on Theoretical Aspects of Computer Science [AF].

that is *encryptable*, in the sense of [AFK], can be used as a building block in a protocol for secure circuit evaluation. We also show how hiding and encryption can be traded off and that it is impossible, in a two-player protocol, for both players to accomplish hiding. Our basic protocol could be extended to work for n players. Because the extended protocol would be complex, and because our basic technique is illustrated by the two-player case, we do not present this extension.

Secure circuit evaluation was first studied by Yao [Y82B]. Since then, many researchers have studied it and related problems, in various models using different techniques [BGW, CCD, CDvdG, GHY87, GMW, H, K, Y86]. Comparisons of previously published protocols for secure circuit evaluation and related problems appear in, e.g., [CDvdG, H]. Before presenting our scheme, we would like to draw attention to the protocol of [CDvdG], which is the one that resembles ours most closely (although it was derived independently). In [CDvdG], Chaum, Damgård, and van de Graaf also give a circuit-evaluation protocol that allows one player to hide his secrets unconditionally; they use a technique called *blinding*, which is similar to what is called hiding information from an oracle in [AFK].[1] The protocol in [CDvdG] is more complex than the one we present, but it also achieves more, e.g., it is an p-player protocol in which each player's secrets are protected against collusion by the other $p-1$.

In Section 2, we present our protocol, after reviewing an essential ingredient from [AFK]. In Section 3, we give our definitions of the terms "hiding" and "encryption" and provide proofs of the security properties of the protocol. Section 4 discusses what types of cheating by players C and D can be prevented. Finally, two open problems are stated in Section 5.

2. The protocol

As a preliminary, we review one of the results in [AFK].[2]

The quadratic residuosity function takes two arguments, an integer k of the form $p \cdot q$, where p and q and distinct primes, approximately the same size, congruent to 3 mod 4, and an integer u in the set $Z_k^*[+1]$ — the integers relatively prime to k with Jacobi symbol 1. (One can define the quadratic residuosity of a broader class of pairs of integers, but the increased generality is not needed here.) The value $r(u,\ k)$ is 1 if there is an integer a in $Z_k^*[+1]$ such that $a^2 \equiv u \bmod k$, and it is 0 otherwise.

In [AFK], we discussed the following scheme for computing the quadratic residuosity function with secret data. To conceal the instance $\langle u,\ k\rangle$, the player chooses b from Z_k^* and c

[1] The term "hiding" is also used in [CDvdG], but there it means something similar to what we mean by the term "encryption."

[2] The scenario considered in [AFK] is one in which player A wishes to know $f(x)$ but lacks the computational resources to compute f. She wants to query an infinitely powerful oracle B and obtain $f(x)$, while hiding x from B. Examples are provided of functions for which it is possible to hide some important information about x, but the main result in [AFK] is negative: A cannot obtain $f(x)$ while revealing only $|x|$ for any f that is NP-hard. In this paper, we consider a more popular scenario, namely one in which the computation of f requires possession of secret algorithms or data, rather than oracular power.

from $\{0,1\}$, both according to uniform distributions, and computes $v = u \cdot b^2 \cdot (-1)^c \bmod k$. Informally, we often denote v by $E(u)$. The new instance is $\langle v,\ k\rangle$, and the key is the pair $\langle b,\ c\rangle$. Let $e = r(u,k)$ and $e' = r(v,k)$. Then, to decode, that is, to compute e from e', just let e equal $e' + c \bmod 2$. We denote the decoded answer by $F(e')$. Correctness follows from the facts that $u \cdot v$ is a residue mod k if and only if both u and v are residues or neither is a residue and that -1 is a quadratic nonresidue if k is of the specified form. Only k needs to be known for all the operations to be feasible in polynomial time. We gave a simple proof in [AFK] that this scheme for encoding instances of the quadratic residuosity problem allows the encoder to obtain $r(u,k)$ without revealing anything about u, in an information-theoretic sense — that is, while hiding u.

This technique for consulting an oracle with no information transfer is a building block in our protocol for secure circuit evaluation. We also use a standard technique for encrypting single bits: given a bit b, it is easy to generate a y such that $r(y,k) = b$ [GM]. Informally, we often denote y by $Y(b)$.

In our current setting, player D has the input x, which can be written $x_1 \ldots x_n$ in binary. Player C has a circuit to compute f on inputs of length n. In the initial phase of the protocol, C sends to D the number a of AND gates in his circuit. D then generates $k = p \cdot q$, where p and q are distinct primes, approximately the same size, congruent to 3 mod 4, and large enough with respect to a, in a sense that we make precise in the next section. Next, D encrypts her input bits $x_1, \ldots, x_n$, using modulus k as explained above. Finally, D sends k and $Y(x_1), \ldots, Y(x_n)$ to C.

We assume without loss of generality that player C's circuit consists of unary NOT gates and binary AND gates and that it has no consecutive NOT gates. In the protocol, C simulates the evaluation of the circuit on D's input. Instead of bits b_i, C will have integers of the form $Y(b_i)$ that represent bits. We must now show how to simulate the logical operations NOT and AND using this scheme for representing bits.

When C must compute the NOT of a bit b, represented by its encryption $Y(b)$, he computes $Y(\overline{b})$. Because $p \equiv q \equiv 3 \bmod 4$, -1 is a quadratic nonresidue mod k, and thus C does not need to know b: $Y(\overline{b}) \equiv (-1) \cdot Y(b) \bmod k$.

To compute the AND of two bits b and b', represented by their encryptions $Y(b)$ and $Y(b')$, C requires the cooperation of D. C transforms b and b' further, to obtain $E(Y(b))$ and $E(Y(b'))$, which he sends to D. Then D computes $b_e = r(E(Y(b)),k)$, $b'_e = r(E(Y(b')),k)$, which she can do efficiently, because she knows p and q. Let b_3, b_2, b_1, and b_0 equal $(b_e \wedge b'_e)$, $(b_e \wedge \overline{b}'_e)$, $(\overline{b}_e \wedge b'_e)$, and $(\overline{b}_e \wedge \overline{b}'_e)$, respectively. D returns $\langle Y(b_3), Y(b_2), Y(b_1), Y(b_0)\rangle$. Since C knows whether $b = b_e$ and whether $b' = b'_e$, he can easily obtain the encrypted conjunction of $Y(b)$ and $Y(b')$ from the message: it is $Y(b_3)$ if both equations hold, $Y(b_2)$ if only the first equation holds, $Y(b_1)$ if only the second equation holds, and $Y(b_0)$ if neither equation holds.

The players follow one of two versions of the final phase of the protocol, depending upon which of them is supposed to receive the answer $f(x)$. In Case 1, C retains the answer. At the end of the gate-simulation phase, C has the encrypted bit $Y(f(x))$. He encodes this bit further by computing $E(Y(f(x)))$, which he then sends to D. Player D returns the residuosity bit $r(E(Y(f(x))),k)$. C decodes the bit to obtain $f(x)$. In Case 2,

in which player D receives the answer, C sends to D the encrypted bit $Y(1) \cdot Y(f(x))$, and D decrypts to obtain the residuosity bit $r(Y(1) \cdot Y(f(x)), k)$, which is, by definition, $f(x)$. As discussed below, in Section 3, C multiplies the encrypted bit $Y(f(x))$ by a random square $Y(1)$ so that D does not learn whether the final gate of the circuit is a NOT or an AND.

In the following description of the protocol, the notation "P1 $\rightarrow$ P2: m" means player P1 sends the message m to player P2. The notation "P: s" means that player P evaluates the statement s.

Initial Phase:

C $\rightarrow$ D: The number of AND gates in his circuit.

D: Generate k and $Y(x_1), \ldots, Y(x_n)$.

D $\rightarrow$ C: $k, Y(x_1), \ldots, Y(x_n)$.

Gate-Simulation Phase:

NOT gate with input $Y(b)$

C: $Y(\overline{b}) := (-1) \cdot Y(b) \bmod k$.

AND gate with inputs $Y(b)$, $Y(b')$

C $\rightarrow$ D: $E(Y(b)), E(Y(b'))$.

D: $b_e := r(E(Y(b)), k)$; $b'_e := r(E(Y(b')), k)$.

$\langle b_3, b_2, b_1, b_0 \rangle := \langle b_e \wedge b'_e,\ b_e \wedge \overline{b}'_e,\ \overline{b}_e \wedge b'_e,\ \overline{b}_e \wedge \overline{b}'_e \rangle$.

D $\rightarrow$ C: $Y(b_3), Y(b_2), Y(b_1), Y(b_0)$.

C: If $b = b_e$ and $b' = b'_e$ then $Y(b \wedge b') := Y(b_3)$.

If $b = b_e$ and $b' \neq b'_e$ then $Y(b \wedge b') := Y(b_2)$.

If $b \neq b_e$ and $b = b'_e$ then $Y(b \wedge b') := Y(b_1)$.

If $b \neq b_e$ and $b' \neq b'_e$ then $Y(b \wedge b') := Y(b_0)$.

Final Phase:

Case 1: C keeps the answer

C $\rightarrow$ D: $E(Y(f(x)))$.

D: $b := r(E(Y(f(x))), k)$.

D $\rightarrow$ C: b.

C: $f(x) := F(b)$.

Case 2: D receives the answer

C $\rightarrow$ D: $Y(1) \cdot Y(f(x))$.

D: $f(x) := r(Y(1) \cdot Y(f(x)), k)$.

It is clear from this description that $f(x)$ is computed correctly. In Case 1, the fact that C computes $f(x)$ follows from the properties of the decoding function F; see [AFK, Theorem 1]. In Case 2, if $f(x) = 1$, then the message $Y(1) \cdot Y(f(x))$ is a quadratic residue mod k, and, if $f(x) = 0$, then $Y(1) \cdot Y(f(x))$ is a quadratic nonresidue. Thus, D computes $f(x)$.

It is also clear from this description that the distinction between "data" and "circuits" is unnecessary. If C has the ability to hide a circuit, then he can also hide some private data, simply by "hardwiring" it into the circuit. Conversely, in protocols in which C has the ability to hide data, he can also hide a circuit through a detour: C can run the protocol, take the circuit for f to be a universal circuit, and use an encoding of the circuit he wants to hide as input.

3. Security properties

Let us first clarify the difference between hiding and encryption. For a general discussion of hiding, see [AFK, BF]; here we only discuss a special case that pertains to two-player protocols.

Let P_0 and P_1 be the two players. Player P_j starts each execution of the protocol with a secret s_j (which in this paper is either the circuit or the input data) and, during the course of the execution, uses a sequence $\overline{c_j}$ of coin tosses. We say that player P_j *hides* a piece of information i if the sequence of messages that P_j sends during the execution is independent of i, s_{1-j}, and $\overline{c_{1-j}}$. Similarly, P_j hides everything about i except l if, given l, the sequence of messages sent by P_j is independent of i, s_{1-j}, and $\overline{c_{1-j}}$. For example, P_j may hide everything about s_j except $|s_j|$ or may hide one variable in the pair $\langle u_j, v_j \rangle$ and not the other. Note that our definition allows l to depend on information possessed by both P_j and P_{1-j}.

Often the sequence of messages sent by P is dependent on the piece of information i, and thus i is not hidden, but this dependence is hard to detect and to exploit in polynomial time. Let Q be another player who interacts with P in a two-party protocol, and, at the end of the protocol, performs a random-polynomial-time computation and outputs a value. The sequence of messages exchanged by P and Q, together with the value output by Q, is called the *transcript* of the protocol. For any particular execution, the transcript and the sequence of private computations performed by Q is called Q's *view* of the protocol. Player P *encrypts* a piece of information i, except for $l(i)$, if, for any random-polynomial-time Q^* who plays the role of Q in the protocol, there is a random-polynomial-time *simulator* M_{Q^*} that, given $l(i)$, can produce a distribution of views that is polynomial-time indistinguishable from the real distribution produced by P and Q^*. Intuitively, whatever knowledge Q^* extracts from the transcript can also be obtained by M_{Q^*} without looking at the transcript. See, for example, [GM, GMR, GMW, H, Y82A] for a discussion of polynomial-time indistinguishability and minimum-knowledge proof systems.

Our protocol uses both hiding and encryption. In this section, we show that C hides everything about his circuit, except the number of AND gates and the modulus k, and

that D encrypts x, except for $|x|$. We also show that there is no two-player protocol in which both players hide everything except one bit (such as the functional value $f(x)$).

Before presenting our proofs, we address the question of why it makes sense to construct a protocol that uses both hiding and encryption. In other words, if one believes the QRA (and in fact relies on it), then why bother to hide some secrets unconditionally? First, as demonstrated by our protocol and that of [CDvdG], hiding can be conceptually simpler than encryption, and it is not necessarily more expensive in terms of computational resources. Second, there may be a disparity in power among players. We may want to treat only certain players as though they could crack cryptosystems based on intractability assumptions. Finally, there may be a disparity in the sensitivity of different secrets. For instance, some data are so ephemeral that, by the time they are decrypted, they are no longer valuable. On the other hand, a circuit (or a secret algorithm) may be used repeatedly over a long period.

For the remainder of this section, a is the number of AND gates in C's circuit and $n = |x|$ is the number of input bits.

Lemma 1: *During the initial and gate-simulation phases,* C *hides everything about his circuit except a and k.*

Proof: The sequence of messages sent from C to D during the first two phases of the protocol is of the form

$$\langle a, y_1, y_2, \ldots, y_{2a-1}, y_{2a} \rangle,$$

where each y_i is a member of $Z_k^*[+1]$; the y_i's are the encoded integers $E(Y(b))$. The distribution of the subsequence y_1, y_2, ..., y_{2a} is uniform on $(Z_k^*[+1])^{2a}$ and, given a and k, the subsequence is independent of the circuit, the input x, and the coin tosses of D. This follows immediately from the structure of the protocol and the fact that the encoding function E hides the integer $Y(b)$ [Theorem 1, AFK]. ∎

In the protocol, D's data is not hidden from C, but it is encrypted. As usual, the proof of polynomial-time indistinguishability depends on a hypothesis about the intractability of a computational problem. Here, the hypothesis we use is the Quadratic Residuosity Assumption (QRA). Intuitively, the QRA says that $r(u, k)$ cannot be computed efficiently, where $k = p \cdot q$, the primes p and q are distinct, approximately the same size, and congruent to 3 mod 4, and $u \in Z_k^*[+1]$. Various versions of the QRA are used extensively in recent cryptographic literature; some attention is devoted in [Y82A] to the relative strengths of the different versions. In what follows, we assume that the pairs (u, k) have the specified form and refer to the computation of $r(u, k)$ as the Quadratic Residuosity Problem (QRP); we use the term "family of polynomial-sized circuits" as it is used throughout the related literature (see, e.g., [H] for details).

QRA: Let $\{C_m\}$ be an arbitrary family of polynomial-sized circuits with a source of random bits. Let $P(C_m)$ denote the probability that C_m outputs the correct bit $r(u, k)$ when given as input a random QRP instance $\langle u, k \rangle$ of length m. Then, for any positive constant d, the inequality

$$P(C_m) < \frac{1}{2} + \frac{1}{m^d}$$

holds for all sufficiently large m.

Some of the literature states this assumption by saying that the QRP gives rise to a *hard-bit family*.

Lemma 2: *Assume that* $|k| > \max(a, n)$. *Then, under the* QRA, D *encrypts* x, *except for* n, *during the initial and gate-simulation phases of the protocol.*

Proof: We show that the distribution of transcripts produced by D and any random-polynomial-time C* can be simulated by a random-polynomial-time machine M_{C^*} with input n. The simulator produces sequences of the form

$$\langle a, k, y_1, \ldots, y_n, z_1, z_2, y_{n+1}, y_{n+2}, y_{n+3}, y_{n+4}, \ldots,$$

$$z_{2a-1}, z_{2a}, y_{n+4a-3}, y_{n+4a-2}, y_{n+4a-1}, y_{n+4a}, o\rangle$$

according to the following distribution:

(1) a is chosen by M_{C^*} exactly as it was chosen by C*,

(2) k is chosen as D chooses it in the initial phase,

(3) the subsequence $\langle y_1, \ldots y_n\rangle$ is drawn uniformly from $(Z_k^*[+1])^n$,

(4) each of the subsequences $\langle y_{n+4i+1}, y_{n+4i+2}, y_{n+4i+3}, y_{n+4i+4}\rangle$, $0 \leq i \leq a-1$, is drawn independently from the uniform distribution on $(Z_k^*[+1])^4$,

(5) each of the subsequences $\langle z_{2i+1}, z_{2i+2}\rangle$, $0 \leq i \leq a-1$, is computed exactly as C* computes his i^{th} message of the gate-simulation phase, given the history $\langle a, k, y_1, \ldots, y_{n+4i}\rangle$, and

(6) o is computed exactly as C* computes his output. (Note that o is the output of C*, not the output of the protocol, as computed in the final phase.)

The QRA guarantees that the distribution of subsequences $\langle y_1, \ldots, y_n, y_{n+1}, \ldots, y_{n+4a}\rangle$ is polynomial-time indistinguishable from the distribution really computed by D. Similarly, under the QRA, the results of the computations performed with the simulated inputs are indistinguishable from those of the computations with the real inputs, because C* is limited to random polynomial time. Together, these observations imply that the views produced by M_{C^*} are polynomial-time indistinguishable from the real views produced by C* and D. ∎

In practice, during the second step of the initial phase, D would choose p and q large enough so that their product k could not be factored in a reasonable amount of time using the best known factoring algorithms. For the purpose of proving that the QRA implies the secrecy of D's input bits, we need to assume that the size of the modulus k is "big enough" with respect to a and n; the requirement that $|k| > \max(a, n)$ suffices. It would also suffice to require that $|k|$ be "polynomially related" to a and n; we chose the statement $|k| > \max(a, n)$ for simplicity and clarity.

We have the following results about the information communicated during the final phase.

Lemma 3: *In* Case 1 *of the final phase,* C *hides everything about his circuit except k, and* D *hides everything about x except k and $f(x)$.*

Proof: The pair of messages exchanged in Case 1 is of the form $\langle y, r(y,k)\rangle$. The distribution of y is uniform on $Z_k^*[+1]$ and, given k, is independent of the circuit, x, and the coin tosses of D. Given k, the one-bit message $r(y,k)$ that D sends still depends on the coin tosses of C, in the following way: half of the possible coin-toss sequences correspond to $f(x) = r(y,k)$ and half to $f(x) = 1 - r(y,k)$. However, given k and $f(x)$, the message $r(y,k)$ is independent of x, the circuit, and C's coin tosses. ∎

We omit the proof of Lemma 4, because it is very similar to that of Lemma 3.

Lemma 4: *In* Case 2 *of the final phase,* D *(obviously) hides everything about x, and* C *hides everything about his circuit except k and $f(x)$.*

In Case 2, C multiplies the encrypted output of the circuit's final gate by a random square so that D does not learn whether the final gate is a NOT or an AND. Suppose that C did not multiply by a random square but instead sent the output of the final gate, that is, $Y(f(x))$. If the final gate is a NOT, then $Y(f(x))$ is the additive inverse, mod k, of one of the elements of some quadruple that D sent during the protocol. If the final gate is an AND, then $Y(f(x))$ is itself one of the elements of the last quadruple that D sent. The probability that $Y(f(x))$ is both of these is exponentially small; thus translation by a random square is necessary in order to conceal whether the final gate is a NOT.

The security properties proven in Lemmas 1-4 are summarized in the following theorem.

Theorem 1: *During the inital and gate-simulation phases of the protocol,* C *hides everything except the modulus k and the number of* AND *gates in the circuit, and (under* QRA) D *encrypts the input except for its size. In* Case 1 *of the final phase,* C *hides everything except k and* D *hides everything except k and $f(x)$. In* Case 2, D *hides everything and* C *hides everything except k and $f(x)$.*

As we remarked in Section 2, it is possible to transform the protocol so that the circuit is encrypted and the data are hidden. To accomplish this, C would use a universal circuit U_n, and modify U_n so that the bits of x are hardwired into it. D would supply a circuit for f as encrypted input.

Both versions of our protocol (the one that hides the circuit while encrypting the data and the one that hides the data while encrypting the circuit) have the property that exactly one bit of information is exchanged; from it, one of the parties computes the value $f(x)$. The encoding of the bit exchanged is a function of the circuit, the input, and the random coin tosses of the parties. Is it possible to construct a two-player protocol that *hides* everything while exchanging the one bit $f(x)$ (as it is in the multiplayer case [BGW, CCD])? Intuitively, the answer must be "no," because the player who sends to the other an encoding of the one bit $f(x)$ needs some information in order to compute it. Our next result gives a very simple, formal proof that such protocols do not exist.

Theorem 2: *No protocol that hides all secrets and communicates only one bit of information can compute $f(x)$ accurately for all f and all x.*

Proof: Suppose that such a protocol did exist. Then it would have to work for functions of two arguments, one owned by C and hardwired into the circuit, and one owned by D. Without loss of generality, we may assume that it is D who sends the one bit that is communicated. We show that there are inputs on which C computes the wrong answer. The crucial fact is that D must decide whether to send a 0 or a 1 based only on her input and on the number a of AND gates in C's circuit.

Consider the boolean function $x = w$, where x is D's input, w is C's input, and $|x| = |w| = n$. Assume without loss of generality that x_1 and x_2 are distinct input strings of length n for which there is a positive probability that D communicates a 0 for functions with a AND gates. There are runs of the protocol on which C receives the same information for D's inputs x_1 and x_2, and thus there is a positive probability that he computes the wrong answer for at least one input when $w = x_1$.

This example of the string-equality function shows that there is a positive probability that C computes a wrong answer for at least one input. We now show that, in fact, if D communicates only one bit of information, then there are functions for which there is a substantial probability that C computes the wrong answer on many inputs. For this proof, we consider the boolean function $x \geq w$, where x and w are as above. Once again, fix a number n of input bits and a number a of AND gates. Let S be the set of inputs x for which the probability that D sends a 0 is at least $1/2$. We can assume without loss of generality that $|S| \geq 2^{n-1}$ and that the probability that C outputs 0, given that D sends 0, is at least $1/2$. Let w be a string for which there are at least 2^{n-2} elements of S that are greater than or equal to w and at least 2^{n-2} elements of S that are less than w. Call these sets S_1 and S_2, respectively. For each $x \in S_1$, $f(x) = 1$, and, for each $x \in S_2$, $f(x) = 0$. Thus, for at least $1/4$ of all possible values of x (those in S_1), the probability that C computes the wrong answer on any particular run of the protocol is

$$\begin{aligned} & P(\text{C outputs } 0) \\ \geq\ & P(\text{D sends } 0 \text{ and C outputs } 0) \\ =\ & P(\text{D sends } 0) \cdot P(\text{C outputs } 0 \mid \text{D sends } 0) \\ \geq\ & 1/4. \quad \blacksquare \end{aligned}$$

Note that the "information-theoretic" security achieved by the protocols in [BGW, CCD] does not contradict Theorem 2. The results in [BGW, CCD] apply only when the number p of players is strictly greater than two and the number of cheaters is strictly less than $p/2$; our Theorem 2 applies only to the case $p = 2$.

4. Cheating

In this section, we address the question of what happens when either C or D tries to cheat. It is common to provide mechanisms to avoid or to detect cheating in protocols for secure computation, e.g., to turn protocols into "validated protocols" in which players use zero-knowledge subprotocols to prove that they have acted honestly (e.g., [GHY87,

GMW]). However, the issue of cheating is different in a setting in which some secrets are hidden rather than encrypted.

How might C try to cheat? He could try to choose the sequence of AND computations that he requests so as to compromise D's data rather than to compute $f(x)$. However, under the QRA, everything that she sends him during the gate-simulation phase of the protocol is something that he could generate himself – quadruples of elements of $Z_k^*[+1]$, three of which are nonresidues and one of which is a residue. In Case 1 of the final phase, she "opens" one bit, allowing him to learn $f(x)$. Thus, by using a different circuit, he could obtain a different boolean function of x, but nothing else. Because Case 1 of the final phase involves the opening of one bit, C could, instead, cheat by obtaining the value $r(u,k)$ for one arbitrary u. To prevent this, one can use a protocol that requires C to "commit" to one circuit and "prove" that he simulated the circuit to which he committed himself; see, e.g., [CDvdG] for a good discussion of this issue.

How might D try to cheat? We cannot accuse D of sending incorrect encrypted bits, because x is D's private data. D could deviate from the protocol by sending wrong answers when C asks her for the AND of two encrypted bits or by sending the wrong residuosity bit in Case 1 of the final phase. This could cause C to compute a wrong answer, but it would not compromise the privacy of C's secrets. C is capable of hiding; that is, none of the messages that he sends to D in the course of the computation conveys any information at all, even when D leads the computation astray. Therefore, D cannot cheat to "decrypt" C's messages and learn something she is not supposed to know.

In any case, D can be required to prove that she computes the (encrypted) output of the AND gates correctly and that she sends the correct residuosity bit in the final phase. The set of tuples $\langle k,\ Y(b),\ Y(b'),\ Y(b_3),\ Y(b_2),\ Y(b_1),\ Y(b_0)\rangle$ that satisfy the following conditions is an NP set: $k = p \cdot q$; $Y(b)$ and $Y(b')$ are legal encryptions of bits; and $\langle Y(b_3),\ Y(b_2),\ Y(b_1),\ Y(b_0)\rangle$ is a legal reply from D to C's query about the AND of b and b'. Using trivial modifications of interactive proof systems in [BCC, GHY89], D can prove that her replies are correct with a zero-knowledge subprotocol. (More efficient subprotocols can be constructed, using the special properties of quadratic residues; we omit discussion of efficiency, because our goal is simply to point out that this type of cheating can be thwarted.) Similarly, in the final phase, D can prove that she knows a certificate of the residuosity of $E(Y(f(x)))$; that is, if $b = 1$, D must prove that she knows a number u such that $u^2 \equiv E(Y(f(x))) \bmod k$, and, if $b = 0$, she must prove that she knows a number u such that $(-1) \cdot u^2 \equiv E(Y(f(x))) \bmod k$.

5. Open problems

Our protocol uses the quadratic residuosity predicate $r(u,k)$ as a building block. No communication is required for C to compute the negation of an encrypted bit, but communication is required for C to compute a conjunction. Can C, by computing some function of $Y(b)$ and $Y(b')$, take the AND of two encrypted bits without asking for help from D? Brassard and Crépeau raised a similar question in [BC], where they used a circuit-evaluation protocol based on "permuted truth tables" in their work on zero-knowledge proof systems.

The quadratic residuosity predicate is not the only one that can be used as a building block in a protocol for secure circuit evaluation. To be usable as a building block, a boolean function must be encryptable (in the sense of [AFK]), be computable by player D (perhaps with knowledge of a secret key), and not be computable by player C. Is it possible to design a better protocol using a different building block? More specifically, we would like to reduce the number of rounds of communication needed by each run of the protocol, or, alternatively, to prove a nontrivial lower bound on the number of rounds needed for secure circuit evaluation.

6. Acknowledgements

We are grateful to Gilles Brassard, Stuart Haber, Cynthia Hibbard, and two anonymous referees for their comments on previous versions of this paper.

7. References

[AF] Martín Abadi and Joan Feigenbaum. "A Simple Protocol for Secure Circuit Evaluation," STACS '88 Proceedings, R. Cori and M. Wirsing (eds.), Springer Verlag (pub.), 1988, 264–272.

[AFK] Martín Abadi, Joan Feigenbaum, and Joe Kilian. "On Hiding Information from an Oracle," J. Comput. System Sci. (39), 1989, 21–50.

[BF] Donald Beaver and Joan Feigenbaum. "Instance Hiding in Multioracle Queries," to appear in STACS '90 Proceedings, C. Choffrut and T. Lengauer (eds.), Springer Verlag (pub.), 1990.

[BGW] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," Proceedings of the 20^{th} Annual ACM Symposium on Theory of Computing, 1988, 1–10.

[BCC] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum Disclosure Proofs of Knowledge," J. Comput. System Sci. (37), 1988, 156–189.

[BC] Gilles Brassard and Claude Crépeau. "Zero-knowledge Simulation of Boolean Circuits," CRYPTO '86 Proceedings, Andrew Odlyzko (ed.), Springer Verlag (pub.), 1987, 223–233.

[CCD] David Chaum, Claude Crépeau, and Ivan Damgård. "Multiparty Unconditionally Secure Protocols," Proceedings of the 20^{th} Annual ACM Symposium on Theory of Computing, 1988, 11–19.

[CDvdG] David Chaum, Ivan Damgård, and Jeroen van de Graaf. "Multiparty Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output," CRYPTO'87 Proceedings, Carl Pomerance (ed.), Springer-Verlag (pub.), 1988, 87–119.

[GHY89] Zvi Galil, Stuart Haber, and Moti Yung. "Minimum-Knowledge Interactive Proofs for Decision Problems," SIAM J. on Comput. (18), 1989, 711–739.

[GHY87] Zvi Galil, Stuart Haber, and Moti Yung. "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model," CRYPTO'87 Proceedings, Carl Pomerance (ed.), Springer-Verlag (pub.), 1988, 135–155.

[**GMW**] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play ANY Mental Game," Proceedings of the 19$^{\text{th}}$ Annual ACM Symposium on Theory of Computing, 1987, 218–229.

[**GM**] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption," J.Comput. System Sci. (28), 1984, 270–299.

[**GMR**] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems," SIAM J. on Comput. (18), 1989, 186–208.

[**H**] Stuart Haber. "Multi-Party Cryptographic Computation: Techniques and Applications," PhD Thesis, Columbia University Computer Science Department, 1988.

[**K**] Joe Kilian. "Founding Cryptography on Oblivious Transfer," Proceedings of the 20$^{\text{th}}$ Annual ACM Symposium on Theory of Computing, 1988, 20–31.

[**Y82A**] Andrew C. Yao. "Theory and Applications of Trapdoor Functions," Proceedings of the 23$^{\text{rd}}$ Annual IEEE Symposium on Foundations of Computer Science, 1982, 80–91.

[**Y82B**] Andrew C. Yao. "Protocols for Secure Computations," Proceedings of the 23$^{\text{rd}}$ Annual IEEE Symposium on Foundations of Computer Science, 1982, 160–164.

[**Y86**] Andrew C. Yao. "How to Generate and Exchange Secrets," Proceedings of the 27$^{\text{th}}$ Annual IEEE Symposium on Foundations of Computer Science, 1986, 162–167.