Covert Two-Party Computation

Luis von Ahn[1], Nicholas J. Hopper[1], and John Langford[2]

[1] Carnegie Mellon University
[2] Toyota Technological Institute

Abstract. We introduce the novel concept of *covert two-party computation.* Whereas ordinary secure two-party computation only guarantees that no more knowledge is leaked about the inputs of the individual parties than the result of the computation, covert two-party computation employs steganography to yield the following additional guarantees: (A) no outside eavesdropper can determine whether the two parties are performing the computation or simply communicating as they normally do; (B) before learning $f(x_A, x_B)$, neither party can tell whether the other is running the protocol; (C) after the protocol concludes, each party can only determine if the other ran the protocol insofar as they can distinguish $f(x_A, x_B)$ from uniformly chosen random bits. Covert two-party computation thus allows the construction of protocols that return $f(x_A, x_B)$ only when it equals a certain value of interest (such as "Yes, we are romantically interested in each other") but for which *neither party can determine whether the other even ran the protocol whenever $f(x_A, x_B)$ does not equal the value of interest.* We introduce security definitions for covert two-party computation and we construct protocols with provable security based on the Decisional Diffie-Hellman assumption.

Keywords: Secure two-party computation, Steganography, Fairness.

1 Introduction

Secure two-party computation allows Alice and Bob to evaluate a function of their secret inputs so that neither learns anything other than the output of the function. A real-world example that is often used to illustrate the applications of this primitive is when Alice and Bob wish to determine if they are romantically interested in each other. Secure two-party computation allows them to do so without revealing their true feelings *unless they are both attracted*. By securely evaluating the AND of the bits representing whether each is attracted to the other, both parties can learn if there is a match without risking embarrassment: if Bob is not interested in Alice, for instance, the protocol does not reveal whether Alice is interested in him. So goes the example.

However, though often used to illustrate the concept, this example is not entirely logical. The very use of two-party computation already reveals possible interest from one party: "would you like to determine if we are both attracted to each other?"

A similar limitation occurs in a variety of other applications where the very use of the primitive raises enough suspicion to defeat its purpose. To overcome this limitation we introduce *covert two-party computation*, which guarantees the following (in addition to leaking no additional knowledge about the individual inputs): (A) no outside eavesdropper can determine whether the two parties are performing the computation or simply communicating as they normally do; (B) before learning $f(x_A, x_B)$, neither party can tell whether the other is running the protocol; (C) after the protocol concludes, each party can only determine if the other ran the protocol insofar as they can distinguish $f(x_A, x_B)$ from uniformly chosen random bits. By defining a functionality $g(x_A, x_B)$ such that $g(x_A, x_B) = f(x_A, x_B)$ whenever $f(x_A, x_B) \in Y$ and $g(x_A, x_B)$ is pseudorandom otherwise, covert two-party computation allows the construction of protocols that return $f(x_A, x_B)$ only when it is in a certain set of interesting values Y but for which *neither party can determine whether the other even ran the protocol whenever $f(x_A, x_B) \notin Y$*. Among the many important potential applications of covert two-party computation we mention the following:

– **Dating.** As hinted above, covert two-party computation can be used to properly determine if two people are romantically interested in each other. It allows a person to approach another

and perform a computation hidden in their normal-looking messages such that: (1) if both are romantically interested in each other, they *both* find out; (2) if none or only one of them is interested in the other, neither will be able to determine that a computation even took place. In case both parties are romantically interested in each other, it is important to guarantee that *both* obtain the result. If one of the parties can get the result while ensuring that the other one doesn't, this party would be able to learn the other's input by pretending he is romantically interested; there would be no harm for him in doing so since the other would never see the result. However, if the protocol is fair (either both obtain the result or neither of them does), parties have a deterrence from lying.

- **Cheating in card games.** Suppose two parties playing a card game want to determine whether they should cheat. Each of them is self-interested, so cheating should not occur unless both players can benefit from it. Using covert two-party computation with both players' hands as input allows them to compute if they have an opportunity to benefit from cheating while guaranteeing that: (1) neither player finds out whether the other attempted to cheat unless they can both benefit from it; (2) none of the other players can determine if the two are secretly planning to collude.
- **Bribes.** Deciding whether to bribe an official can be a difficult problem. If the official is corrupt, bribery can be extremely helpful and sometimes necessary. However, if the official abides by the law, attempting to bribe them can have extremely negative consequences. Covert two-party computation allows individuals to approach officials and negotiate a bribe with the following guarantees: (1) if the official is willing to accept bribes and the individual is willing to give them, the bribe is agreed to; (2) if at least one of them is not willing to participate in the bribe, neither of them will be able to determine if the other attempted or understood the attempt of bribery; (3) the official's supervisor, even after seeing the entire sequence of messages exchanged, will not be able to determine if the parties performed or attempted bribery.

Our protocols make use of provably secure steganography [12, 2, 3, 16] to hide the computation in innocent-looking communications. Steganography alone, however, is not enough. Combining steganography with two-party computation in the obvious black-box manner (i.e., forcing all the parties participating in an ordinary two-party protocol to communicate steganographically) yields protocols that are undetectable to an outside observer but does not guarantee that the participants will fail to determine if the computation took place. Depending on the output of the function, we wish to hide that the computation took place even from the participants themselves.

Who Knows What? Given the guarantees that covert-two party computation offers, it is important to clarify what the parties know and what they don't. We assume that both parties know a common circuit for the function that they wish to evaluate, that they know which role they will play in the evaluation, and that they know when to start evaluating the circuit if the computation is going to occur. An example of such "synchronization" information could be: "if we will determine whether we both like each other, the computation will start with the first message exchanged after 5pm." We assume adversarial parties know all such details of the protocols we construct.

Roadmap. The high-level view of our presentation is as follows. First, we define "ordinary" or "innocent-looking" communications. Our protocols will generate messages that are indistinguishable from "ordinary" communications — so nobody can tell if the parties are performing a computation or just communicating innocently. The first protocol we present will be a modification of Yao's "garbled circuit" two-party protocol in which, except for the oblivious transfer, all messages generated are indistinguishable from uniform random bits. We construct a protocol for oblivious transfer that generates messages that are indistinguishable from uniform random bits (under the Decisional Diffie-Hellman assumption) to yield a complete protocol for two-party secure function evaluation that generates messages indistinguishable from random bits. We then use steganography to transform this into a protocol that generates messages indistinguishable from "ordinary" communications. The protocol thus constructed, however, is not secure against malicious adversaries

nor is it fair (since neither is Yao's protocol by itself). We therefore construct another protocol, which uses our modification of Yao's protocol as a subroutine, that satisfies fairness and is secure against malicious adversaries, in the Random Oracle Model. The major difficulty in doing so is that the standard zero-knowledge-based techniques for converting a protocol in the honest-but-curious model into a protocol secure against malicious adversaries cannot be applied in our case, since they reveal that that the other party is running the protocol.

Related Work. Secure two-party computation was introduced by Yao [17]. Since then, there have been several papers on the topic and we refer the reader to a survey by Goldreich [8] for further references. Constructions that yield fairness for two-party computation were introduced by Yao [18], Galil et al. [7], Brickell et al. [5], and many others (see [15] for a more complete list of such references). The notion of covert two-party computation, however, appears to be completely new.

Notation. We say a function $\mu : \mathbb{N} \to [0,1]$ is *negligible* if for every $c > 0$, for all sufficiently large k, $\mu(k) < 1/k^c$. We denote the length (in bits) of a string or integer s by $|s|$ and the concatenation of string s_1 and string s_2 by $s_1||s_2$. We let U_k denote the uniform distribution on k bit strings. If $\mathcal{D}$ is a distribution with finite support X, we define the *minimum entropy* of $\mathcal{D}$ as $H_\infty(\mathcal{D}) = \min_{x \in X}\{\log_2(1/\Pr_{\mathcal{D}}[x])\}$. The *statistical distance* between two distributions $\mathcal{C}$ and $\mathcal{D}$ with joint support X is defined by $\Delta(\mathcal{C},\mathcal{D}) = (1/2)\sum_{x\in X} |\Pr_{\mathcal{D}}[x] - \Pr_{\mathcal{C}}[x]|$. Two sequences of distributions, $\{\mathcal{C}_k\}_k$ and $\{\mathcal{D}_k\}_k$, are called *computationally indistinguishable*, written $\mathcal{C} \approx \mathcal{D}$, if for any probabilistic polynomial-time $\mathbf{A}$, $\mathbf{Adv}^{\mathbf{A}}_{\mathcal{C},\mathcal{D}}(k) = |\Pr[\mathbf{A}(\mathcal{C}_k) = 1] - \Pr[\mathbf{A}(\mathcal{D}_k) = 1]|$ is negligible in k.

2 Bidirectional Channels.

We hide the communication patterns of two-party computation protocols in "ordinary" or "innocent-looking" messages. We define ordinary communication patterns and messages in a manner similar to the *channels* used by [12, 2, 16]. The main difference is that our channel is shared among two participants and messages sent by each participant might depend on previous messages sent by either one of them. To emphasize this difference, we use the term *bidirectional channel*.

Messages are drawn from a set D of *documents*. For simplicity we assume that time proceeds in discrete *timesteps*. Each party $P \in \{P_0, P_1\}$ maintains a history h_P, which represents a timestep-ordered list of all documents sent and received by P. We call the set of well-formed histories $\mathcal{H}$. We associate to each party P a family of probability distributions $\mathcal{B}^P = \left\{\mathcal{B}^P_h\right\}_{h\in\mathcal{H}}$ on D.

The communication over a bidirectional channel $\mathcal{B} = (D, \mathcal{H}, \mathcal{B}^{P_0}, \mathcal{B}^{P_1})$ proceeds as follows. At each timestep, each party P receives messages sent to them in the previous timestep, updates h_P accordingly, and draws a document $d \leftarrow \mathcal{B}^P_{h_P}$ (the draw could result in the empty message $\perp$, signifying that no action should be taken that timestep). The document d is then sent to the other party and h_P is updated. We assume for simplicity that all messages sent at a given timestep are received at the next one. Denote by $\mathcal{B}^P_{h_P} \neq \perp$ the distribution $\mathcal{B}^P_{h_P}$ conditioned on not drawing $\perp$. We will consider families of bidirectional channels $\{\mathcal{B}_k\}_{k\geq 0}$ such that: (1) the length of elements in D_k is polynomially-bounded in k; (2) for each $h \in \mathcal{H}_k$ and party P, either $\Pr[\mathcal{B}^P_h = \perp] = 1$ or $\Pr[\mathcal{B}^P_h = \perp] \leq 1 - \delta$, for constant δ; and (3) there exists a function $\ell(k) = \omega(\log k)$ so that for each $h \in \mathcal{H}_k$, $H_\infty((\mathcal{B}^P_h)_k \neq \perp) \geq \ell(k)$ (that is, there is some variability in the communications).

We assume that party P can draw from $\mathcal{B}^P_h$ for any history h, and that the adversary can draw from $\mathcal{B}^P_h$ for every party P and history h. We assume that the ability to draw from these distributions does not contradict the cryptographic assumptions that our results are based on. In the rest of the paper, all communications will be assumed to conform to the bidirectional channel structure: parties only communicate by sending documents from D to each other and parties not running a protocol communicate according to the distributions specified by $\mathcal{B}$. Parties running a protocol strive to communicate using sequences of documents that appear to come from $\mathcal{B}$. As a

convention, when $\mathcal{B}$ is compared to another random variable, we mean a random variable which draws from the process $\mathcal{B}$ the same number of documents as the variable we are comparing it to.

Bidirectional channels provide a model of the distribution of communications between two parties and are general enough to express almost any form of communication between the parties.

3 Covert Two-Party Computation Against Honest-but-Curious Adversaries

We now present a protocol for covert two-party computation that is secure against honest-but-curious adversaries in the standard model (no Random Oracles) and assumes that the decisional Diffie-Hellman problem is hard. The protocol is based on Yao's well-known function evaluation protocol [17].

We first define covert two-party computation formally, following standard definitions for secure two-party computation, and we then describe Yao's protocol and the necessary modifications to turn it into a covert computation protocol. The definition presented in this section is only against honest-but-curious adversaries and is unfair in that only one of the parties obtains the result. In Section 4 we will define covert two-party computation against malicious adversaries and present a protocol that is fair: either both parties obtain the result or neither of them does. The protocol in Section 4 uses the honest-but-curious protocol presented in this section as a subroutine.

3.1 Definitions

Formally, a two-party, n-round protocol is a pair $\Pi = (P_0, P_1)$ of programs. The computation of Π proceeds as follows: at each round, P_0 is run on its input x_0, the security parameter 1^k, a state s_0, and the (initially empty) history of messages exchanged so far, to produce a new message m and an internal state s_0. The message m is sent to P_1, which is run on its input x_1, the security parameter 1^k, a state s_1, and the history of messages exchanged so far to produce a message that is sent back to P_0, and a state s_1 to be used in the next round. Denote by $\langle P_0(x_0), P_1(x_1)\rangle$ the *transcript* of the interaction of P_0 with input x_0 and P_1 with input x_1. This transcript includes all messages exchanged between P_0 and P_1 along with the timestep in which they were sent. After n rounds, each party $P \in \{P_0, P_1\}$ halts with an output, denoted by $\Pi_P(x_0, x_1) = \Pi_P(\bar{x})$. We say that Π *correctly realizes the functionality* f if for at least one $P \in \{P_0, P_1\}$, $\Pr[\Pi_P(\bar{x}) = f(\bar{x})] \geq 1 - \nu(k)$, where ν is negligible.

For $\sigma \in \{0, 1\}$, we denote by $V_{\Pi}^{P_\sigma}(x_0, x_1)$ the *view* of party P_σ on input x_σ when interacting with $P_{1-\sigma}$ on input $x_{1-\sigma}$. The view includes P_σ's input x_σ, private random bits, and all messages sent by P_0 and P_1. We say Π *securely realizes the functionality* f if Π correctly realizes f and, for any P'_σ and $x_{1-\sigma}$, there is a *simulator* P''_σ and an x_σ such that $P''_\sigma(f(x_0, x_1)) \approx V_{\Pi}^{P'_\sigma}(x_0, x_1)$. Notice that given $f(x_0, x_1)$, P'_σ could just use P''_σ to simulate his interaction with $P_{1-\sigma}$ without actually running Π. Thus if Π securely implements f, neither party learns more from the interaction than could be learned from just $f(x_0, x_1)$.

Define the view of party P interacting in protocol Π up through round j by $V_{\Pi,j}^{P}(\bar{x})$. When party P_σ is not executing Π but is drawing from $\mathcal{B}$ instead, we denote this "protocol" by $\Pi : \mathcal{B}_\sigma$.

Definition 1. *(Covert two-party protocol against honest-but-curious adversaries) We say an n-round, two-party protocol (P_0, P_1)* covertly realizes the functionality f for bidirectional channel $\mathcal{B}$ *if it securely realizes f and if it has the following additional properties:*

1. *(External covertness): For any input $\bar{x}$, $\langle P_0(x_0), P_1(x_1)\rangle \approx \mathcal{B}$.*
2. *(Internal covertness): For any input $\bar{x}$, $V_{\Pi,n}^{P_0}(\bar{x}) \approx V_{\Pi:\mathcal{B}_1,n}^{P_0}(\bar{x})$ and $V_{\Pi,n-1}^{P_1}(\bar{x}) \approx V_{\Pi:\mathcal{B}_0,n-1}^{P_1}(\bar{x})$.*
3. *(Final Covertness): For every PPT D there exists a PPT D' and a negligible ν such that for any x_1 and any distribution X_0, $\mathbf{Adv}^{D}_{V_{\Pi}^{P_1}(X_0,x_1), V_{\Pi:\mathcal{B}_0}^{P_1}(X_0,x_1)}(k) \leq \mathbf{Adv}^{D'}_{f(X_0,x_1),U_l}(k) + \nu(k)$.*

In other words, until the final round, neither party can distinguish between the case that the other is running the protocol or just drawing from $\mathcal{B}$; and after the final message, P_0 still cannot tell, while P_1 can only distinguish the cases if $f(x_0, x_1)$ and U_m are distinguishable. Note that property 2 implies property 1, since P_0 could apply the distinguisher to his view (less the random bits).

We will slightly abuse notation and say that a protocol which has messages indistinguishable from random bits (even given one party's view) is *covert for the uniform channel* $\mathcal{U}$.

3.2 Yao's Protocol For Two-Party Secure Function Evaluation

Yao's protocol [17] securely (not covertly) realizes any functionality f that is expressed as a combinatorial circuit. Our description is based on [14]. The protocol is run between two parties, the *Input Owner* A and the *Program Owner* B. The input of A is a value x, and the input of B is a description of a function f. At the end of the protocol, B learns $f(x)$ (and nothing else about x), and A learns nothing about f. The protocol requires two cryptographic primitives, pseudorandom functions and oblivious transfer, which we describe here for completeness.

Pseudorandom Functions. Let $\{F : \{0,1\}^k \times \{0,1\}^{L(k)} \to \{0,1\}^{l(k)}\}_k$ denote a sequence of function families. Let $\mathbf{A}$ be an oracle probabilistic adversary. We define the *prf-advantage of* $\mathbf{A}$ *over* F as $\mathbf{Adv}^{\text{prf}}_{F,\mathbf{A}}(k) = |\Pr_K[\mathbf{A}^{F_K(\cdot)}(1^k) = 1] - \Pr_g[\mathbf{A}^g(1^k) = 1]|$, where $K \leftarrow U_k$ and g is a uniformly chosen function from $L(k)$ bits to $l(k)$ bits. Then F is *pseudorandom* if $\mathbf{Adv}^{\text{prf}}_{F,\mathbf{A}}(k)$ is negligible in k for all polynomial-time $\mathbf{A}$. We will write $F_K(\cdot)$ as shorthand for $F_{|K|}(K, \cdot)$ when $|K|$ is known.

Oblivious Transfer. 1-out-of-2 oblivious transfer (OT^2_1) allows two parties, the *sender* who knows the values m_0 and m_1, and the *chooser* whose input is $\sigma \in \{0,1\}$, to communicate in such a way that at the end of the protocol the chooser learns m_σ, while learning nothing about $m_{1-\sigma}$, and the sender learns nothing about σ. Formally, let $\mathcal{O} = (S, C)$ be a pair of interactive PPT programs. We say that $\mathcal{O}$ is *correct* if $\Pr[\mathcal{O}_C((m_0, m_1), \sigma) = m_\sigma] \geq 1 - \epsilon(k)$ for negligible ϵ. We say that $\mathcal{O}$ has *chooser privacy* if for any PPT S' and any m_0, m_1, $\left|\Pr[S'(\langle S'(m_0, m_1), C(\sigma)\rangle) = \sigma] - \frac{1}{2}\right| \leq \epsilon(k)$ and $\mathcal{O}$ has *sender privacy* if for any PPT C' there exists a σ and a PPT C'' such that $C''(m_\sigma) \approx V^{C'}_{\Pi}((m_0, m_1), \sigma)$. We say that $\mathcal{O}$ *securely realizes the functionality* OT^2_1 if $\mathcal{O}$ is correct and has chooser and sender privacy.

Yao's Protocol. Yao's protocol is based on expressing f as a combinatorial circuit. Starting with the circuit, the program owner B assigns to each wire i two random k-bit values (W^0_i, W^1_i) corresponding to the 0 and 1 values of the wire. It also assigns a random permutation π_i over $\{0,1\}$ to the wire. If a wire has value b_i we say it has "garbled" value $(W^{b_i}_i, \pi_i(b_i))$. To each gate g, B assigns a unique identifier I_g and a table T_g which enables computation of the garbled output of the gate given the garbled inputs. Given the garbled inputs to g, T_g does not disclose any information about the garbled output of g for any other inputs, nor does it reveal the actual values of the input bits or the output bit.

Assume g has two input wires (i, j) and one output wire *out* (gates with higher fan in or fan out can be accommodated with straightforward modifications). The construction of T_g uses a pseudorandom function F whose output length is $k + 1$. The table T_g is as follows:

$\pi_i(b_i)$	$\pi_j(b_j)$	value
0	0	$(W_{out}^{g(b_i,b_j)}, \pi_o(b_{out})) \oplus F_{W_j^{b_j}}(I_g, 0) \oplus F_{W_i^{b_i}}(I_g, 0)$
0	1	$(W_{out}^{g(b_i,b_j)}, \pi_o(b_{out})) \oplus F_{W_j^{b_j}}(I_g, 0) \oplus F_{W_i^{b_i}}(I_g, 1)$
1	0	$(W_{out}^{g(b_i,b_j)}, \pi_o(b_{out})) \oplus F_{W_j^{b_j}}(I_g, 1) \oplus F_{W_i^{b_i}}(I_g, 0)$
1	1	$(W_{out}^{g(b_i,b_j)}, \pi_o(b_{out})) \oplus F_{W_j^{b_j}}(I_g, 1) \oplus F_{W_i^{b_i}}(I_g, 1)$

To compute $f(x)$, B computes garbled tables T_g for each gate g, and sends the tables to A. Then, for each circuit input wire i, A and B perform an oblivious transfer, where A plays the role of the chooser (with $\sigma = b_i$) and B plays the role of the sender, with $m_0 = W_i^0 \| \pi_i(0)$ and $m_1 = W_i^1 \| \pi_i(1)$. A computes $\pi_j(b_j)$ for each output wire j of the circuit (by trickling down the garbled inputs using the garbled tables) and sends these values to B, who applies π_j^{-1} to learn b_j. Alternatively, B can send the values π_j (for each circuit output wire j) to A, who then learns the result. Notice that the first two columns of T_g can be implicitly represented, leaving a "table" which is indistinguishable from uniformly chosen bits.

3.3 Steganographic Encoding

We use provably secure steganography to transform Yao's protocol into a covert two-party protocol; we also use it as a building block for all other covert protocols presented in this paper. For completeness we state a construction that has appeared in various forms in [6, 12, 2]. Let $\mathcal{HASH}$ denote a family of hash functions $H : D \to \{0,1\}^c$ which is *pairwise independent*, that is, for any $x_1 \neq x_2 \in D$, for any $y_1, y_2 \in \{0,1\}^m$, $\Pr_H[H(x_1) = y_1 \wedge H(x_2) = y_2] = 1/2^{2m}$. Let $\mathcal{D}$ denote an arbitrary probability distribution on D satisfying $H_\infty(\mathcal{D}) = \ell(k)$ where k is the security parameter. The following constructions hide and recover m uniformly-chosen bits in a distribution indistinguishable from $\mathcal{D}$ when $\ell(k) - m = \omega(\log k)$.

Protocol 1 *(Basic steganographic encoding/decoding routines)*

Procedure $\texttt{Basic\_Encode}^{\mathcal{D}}$:
Input: $H \in \mathcal{HASH}$, $c \in \{0,1\}^m$
Let $j = 0$
repeat:
  sample $s \leftarrow \mathcal{D}$, increment j
until $H(s) = c$ OR $(j > k)$
Output: s

Procedure $\texttt{Basic\_Decode}$:
Input: $H \in \mathcal{HASH}$, $s \in D$
set $c = H(s)$
Output: c

Proposition 1. *Let $H \leftarrow \mathcal{HASH}$. Then $\Delta((H, \texttt{Basic\_Encode}^{\mathcal{D}}(H, U_m)), (H, \mathcal{D})) \leq 2^{-(\ell(k)-m)/2+1}$.*

The result follows from the Leftover Hash Lemma ([11], Lemma 4.8). Intuitively, it guarantees that $\texttt{Basic\_Encode}(c)$ will be (statistically) indistinguishable from the messages exchanged in a bidirectional channel whenever c is a uniformly chosen bit string. (When we refer to $\texttt{Basic\_Encode}$ with only a single argument, we implicitly assume that an appropriate h has been chosen and is publicly accessible to all parties.)

Thus, to guarantee covertness for channel $\mathcal{B}$, we will ensure that all our protocols generate messages that are indistinguishable from uniformly chosen random bits and then encode these messages with $\texttt{Basic\_Encode}$. Formally, suppose $\Pi = (P_0, P_1)$ is an arbitrary two-party protocol which securely realizes the functionality f. We will construct a protocol $\Sigma^\Pi = (S_0^{P_0}, S_1^{P_1})$ which has the property that if $V_\Pi^{P_b}(\bar{x})$ is indistinguishable from uniformly chosen bits (that is, Π covertly realizes f for the uniform channel), then Σ^Π covertly realizes the functionality f for channel $\mathcal{B}$. We assume that P_0, P_1 have the property that, given a partial input, they return the string ε, indicating

that more bits of input are needed. Then $S_b^{P_b}$ has the following round function (which simply uses `Basic_Encode` and `Basic_Decode` to encode and decode all messages exchanged by P_0 and P_1):

Protocol 2 *(Transformation to a covert protocol)*

Procedure $S_b^{P_b}$*:*
Input: *history* $h \in \mathcal{H}$, `state`, *document* $s \in D$
draw $d \leftarrow \mathcal{B}_h^{P_b}$
if (`state`.*status* $=$ *"receiving") then*
  set `state`.$msg = $ `state`.$msg \| $`Basic_Decode`$(s)$
  set $c = P_b($`state`.$msg)$
  if $(c \neq \varepsilon)$ *set* `state`.*status* $=$ *"sending"; set* `state`.$msg = c$
if (`state`.*status* $=$ *"sending") then*
  if $(d \neq \perp)$ *then*
   set $c =$ *first* m *bits of* `state`.msg
   set `state`.msg $=$ `state`.msg *without the first* m *bits*
   set $d =$ `Basic_Encode`$^{(\mathcal{B}_h^{P_b} \neq \perp)}(c)$
   if `state`.$msg =$ *""* *set* `state`.*status* $=$ *"receiving"*
Output: *message* d, `state`

Theorem 1. *If* Π *covertly realizes the functionality* f *for the uniform channel, then* Σ^{Π} *covertly realizes* f *for the bidirectional channel* $\mathcal{B}$*.*

Proof. Let k^c be an upper bound on the number of bits in $\langle P_0(x_0), P_1(x_1)\rangle$. Then Σ^{Π} transmits at most $2k^c/m$ (non-empty) documents. Suppose there is a distinguisher D for $V_{\Sigma}^{S_b}(\bar{x})$ from $V_{\Sigma:\mathcal{B}_{1-b}}^{S_b}(\bar{x})$ with significant advantage ϵ. Then D can be used to distinguish $V_{\Pi}^{P_b}(\bar{x})$ from $V_{\Pi:\mathcal{U}_{1-b}}^{P_b}(\bar{x})$, by simulating each round as in Σ to produce a transcript T; If the input is uniform, then $\Delta(T, \mathcal{B}) \leq (k^c/m)2^{2-(\ell(k)-m)/2} = \nu(k)$, and if the input is from Π, then T is identical to $V_{\Sigma}^{S_b}(\bar{x})$. Thus D's advantage in distinguishing Π from $\Pi : \mathcal{U}_{1-b}$ is at least $\epsilon - \nu(k)$.

IMPORTANT: For the remainder of the paper we will present protocols Π that covertly realize f for $\mathcal{U}$. It is to be understood that the final protocol is meant to be Σ^{Π}, and that when we state that "Π covertly realizes the functionality f" we are referring to Σ^{Π}.

3.4 Covert Oblivious Transfer

As mentioned above, we guarantee the security of our protocols by ensuring that all the messages exchanged are indistinguishable from uniformly chosen random bits. To this effect, we present a modification of the protocol in [4] for oblivious transfer that ensures that all messages exchanged are indistinguishable from uniform when the input messages m_0 and m_1 are uniformly chosen. Our protocol relies on the well-known integer decisional Diffie-Hellman assumption:

Integer Decisional Diffie-Hellman. Let P and Q be primes such that Q divides $P-1$, let $\mathbb{Z}_P^*$ be the multiplicative group of integers modulo P, and let $g \in \mathbb{Z}_P^*$ have order Q. Let $\mathbf{A}$ be an adversary that takes as input three elements of $\mathbb{Z}_P^*$ and outputs a single bit. Define the *DDH advantage of* $\mathbf{A}$ *over* (g, P, Q) as: $\mathbf{Adv}_{\mathbf{A}}^{\mathrm{ddh}}(g, P, Q) = |\Pr_{a,b,r}[\mathbf{A}_r(g^a, g^b, g^{ab}, g, P, Q) = 1] - \Pr_{a,b,c,r}[\mathbf{A}_r(g^a, g^b, g^c, g, P, Q) = 1]|$, where $\mathbf{A}_r$ denotes the adversary $\mathbf{A}$ running with random tape r, a, b, c are chosen uniformly at random from $\mathbb{Z}_Q$ and all the multiplications are over $\mathbb{Z}_P^*$. The Integer Decisional Diffie-Hellman assumption (DDH) states that for every PPT $\mathbf{A}$, for every sequence $\{(P_k, Q_k, g_k)\}_k$ satisfying $|P_k| = k$ and $|Q_k| = \Theta(k)$, $\mathbf{Adv}_{\mathbf{A}}^{\mathrm{ddh}}(g_k, P_k, Q_k)$ is negligible in k.

Setup. Let $p = rq+1$ where $2^k < p < 2^{k+1}$, q is a large prime, and $\gcd(r,q) = 1$; let g generate $\mathbb{Z}_p^*$ and thus $\hat{g} = g^r$ generates the unique multiplicative subgroup of order q. Assume $|m_0| = |m_1| < k/2$. Let $H : \{0,1\}^{2k} \times \mathbb{Z}_p \to \{0,1\}^{k/2}$ be a pairwise-independent family of hash functions.

Protocol 3 *COT:*

S: **Input:** m_0, m_1		C: **Input:** σ
chooses $c \leftarrow \mathbb{Z}_p^*$ *subject to* $c \leq 2^k$.		
	$\xrightarrow{\quad c \quad}$	*chooses* $\alpha \in \mathbb{Z}_{p-1}$ *subject to* $h_0 \leq 2^k$, *where* $h_\sigma = g^\alpha \bmod p$ *and* $h_{1-\sigma} = c/g^\alpha \bmod p$.
chooses $r_0, r_1 \in \mathbb{Z}_{p-1}$ *subject to* $g_b = g^{r_b} \leq 2^k$, *for* $b \in \{0,1\}$.	$\xleftarrow{\quad h_0 \quad}$	
chooses $y_0, y_1 \leftarrow U_{2k}$.		
sets $C_b = H(y_b, h_b^{r_b r}) \oplus m_b$, $b \in \{0,1\}$.		
	$\xrightarrow{\quad g_0 \| y_0 \| C_0 \| g_1 \| y_1 \| C_1 \quad}$	*computes* $m_\sigma = H(y_\sigma, g_\sigma^{r\alpha}) \oplus C_\sigma$.

Lemma 1. *S cannot distinguish between the case that C is following the COT protocol and the case that C is drawing from U_k; that is, $V_{COT}^S(m_0, m_1, \sigma) \approx V_{COT:\mathcal{U}_C}^S(m_0, m_1, \sigma)$.*

Proof. This follows because the message sent by C is uniformly chosen.

Lemma 2. *When $m_0, m_1 \leftarrow U_{k/2}$, C cannot distinguish between the case that S is following the COT protocol and the case that S is sending uniformly chosen strings. That is, $V_{COT}^C(m_0, m_1, \sigma) \approx V_{COT:\mathcal{U}_S}^C(m_0, m_1, \sigma)$.*

Proof. When m_0, m_1 are uniform, all of the messages generated by S are uniformly distributed.

Lemma 3. *The COT protocol securely realizes the OT_1^2 functionality.*

Proof. Clearly, S learns nothing about the value σ, since h_0 is identically distributed in both cases $C(c,0)$ and $C(c,1)$. Now, suppose without loss of generality that $\sigma = 0$. We will argue that for any (m_0, m_1, m_1'), the messages generated by $S(m_0, m_1, h_0)$ and $S(m_0, m_1', h_0)$ are indistinguishable to C, thus proving that C learns nothing about m_1. Consider two hybrid COT protocols: COT1 uniformly chooses a value $r_2 \in \mathbb{Z}_q$ and replaces $h_1^{r_1 r}$ with $\hat{g}^{r_2}$, while COT2 replaces the value $H(y_1, h_1^{r_1 r})$ with a uniformly chosen $k/2$-bit string. Clearly, $S_{COT2}(m_0, m_1, h_0)$ and $S_{COT2}(m_0, m_1', h_0)$ are indistinguishable. But the Leftover-Hash lemma implies that $S_{COT2}(x)$ and $S_{COT1}(x)$ are indistinguishable. The Decisional Diffie-Hellman assumption implies that $S_{COT1}(x)$ and $S(x)$ are indistinguishable, since a distinguisher between $S_{COT1}(x)$ and $S(x)$ with advantage ϵ can be used to distinguish between $(\hat{g}^a, \hat{g}^b, \hat{g}^{ab})$ and $(\hat{g}^a, \hat{g}^b, \hat{g}^c)$ with advantage at least $\epsilon/4$. This can be done using the technique from [2] of converting elements of the subgroup of order Q into elements of $\mathbb{Z}_p^*$ using the Chinese Remainder Theorem in the exponent, and aborting if either conversion results in an element greater than 2^k (which happens with probability at most $\frac{3}{4}$). Thus it must be that $S(m_0, m_1, h_0)$ and $S(m_0, m_1', h_0)$ are indistinguishable.

Conjoining these three lemmas gives the following theorem:

Theorem 2. *Protocol COT covertly realizes the uniform-OT_1^2 functionality*

3.5 Combining The Pieces

We can combine the components developed up to this point to make a protocol which covertly realizes any two-party functionality. The final protocol, which we call COVERT-YAO, is simple: assume that both parties know a circuit C_f computing the functionality f. Bob first uses Yao's protocol to create a garbled circuit for $f(\cdot, x_B)$. Alice and Bob perform $|x_A|$ covert oblivious transfers for the garbled wire values corresponding to Alice's inputs. Bob sends the garbled gates to Alice. Finally, Alice collects the garbled output values and sends them to Bob, who de-garbles these values to obtain the output.

Theorem 3. *The* COVERT-YAO *protocol covertly realizes the functionality f.*

Proof. That (Alice, Bob) securely realize the functionality f follows from the security of Yao's protocol. Now consider the distribution of each message sent from Alice to Bob:

- In each execution of COT: each message sent by Alice is uniformly distributed
- Final values: these are masked by the uniformly chosen bits that Bob chose in garbling the output gates. To an observer, they are uniformly distributed.

Thus Bob's view, until the last round, is in fact identically distributed when Alice is running the protocol and when she is drawing from $\mathcal{U}$. Likewise, consider the messages sent by Bob:

- In each execution of COT: because the W_i^b from Yao's protocol are uniformly distributed, Theorem 2 implies that Bob's messages are indistinguishable from uniform strings.
- When sending the garbled circuit, the pseudorandomness of F and the uniform choice of the W_i^b imply that each garbled gate, even given one garbled input pair, is indistinguishable from a random string.

Thus Alice's view after all rounds of the protocol is indistinguishable from her view when Bob draws from $\mathcal{U}$.

If Bob can distinguish between Alice running the protocol and drawing from $\mathcal{B}$ after the final round, then he can also be used to distinguish between $f(X_A, x_B)$ and U_l. The approach is straightforward: given a candidate y, use the simulator from Yao's protocol to generate a view of the "data layer." If $y \leftarrow f(X_A, x_B)$, then, by the security of Yao's protocol, this view is indistinguishable from Bob's view when Alice is running the covert protocol. If $y \leftarrow U_l$, then the simulated view of the final step is distributed identically to Alice drawing from $\mathcal{U}$. Thus Bob's advantage will be preserved, up to a negligible additive term.

Notice that as the protocol COVERT-YAO is described, it is not secure against a malicious Bob who gives Alice a garbled circuit with different operations in the gates, which could actually output some constant message giving away Alice's participation even when the value $f(x_0, x_1)$ would not. If instead Bob sends Alice the masking values for the garbled output bits, Bob could still prevent Alice from learning $f(x_0, x_1)$ but could not detect her participation in the protocol in this way. We use this version of the protocol in the next section.

4 Fair Covert Two-party Computation Against Malicious Adversaries

The protocol presented in the previous section has two serious weaknesses. First, because Yao's construction conceals the function of the circuit, a malicious Bob can garble a circuit that computes some function other than the result Alice agreed to compute. In particular, the new circuit could give away Alice's input or output some distinguished string that allows Bob to determine that Alice is running the protocol. Additionally, the protocol is *unfair:* either Alice or Bob does not get the result.

In this section we present a protocol that avoids these problems. In particular, our solution has the following properties: (1) If both parties follow the protocol, both get the result; (2) If Bob

cheats by garbling an incorrect circuit, neither party can tell whether the other is running the protocol, except with negligible advantage; and (3) Except with negligible probability, if one party terminates early and computes the result in time T, the other party can compute the result in time at most $O(T)$. Our protocol is secure in the random oracle model, under the Decisional Diffie Hellman assumption. We show at the end of this section, however, that our protocol can be made to satisfy a slightly weaker security condition without the use of a random oracle. (We note that the technique used in this section has some similarities to one that appears in [1].)

4.1 Definitions

We assume the existence of a non-interactive bitwise commitment scheme with commitments which are indistinguishable from random bits. One example is the (well-known) scheme which commits to b by $commit(b;(r,x)) = r\|\pi(x)\|(x\cdot r)\oplus b$, where π is a one-way permutation on domain $\{0,1\}^k$, $x\cdot y$ denotes the inner-product of x and y over $GF(2)$, and $x, r \leftarrow U_k$. The integer DDH assumption implies the existence of such permutations.

Let f denote the functionality we wish to compute. We say that f is *fair* if for every distinguisher D_σ distinguishing $f(X_0, X_1)$ from U given X_σ with advantage at least ϵ, there is a distinguisher $D_{1-\sigma}$ with advantage at least $\epsilon - \nu(k)$, for a negligible function ν. (That is, if P_0 can distinguish $f(X_0, X_1)$ from uniform, so can P_1.) We say f is *strongly fair* if $(f(X_0, X_1), X_0) \approx (f(X_0, X_1), X_1)$.

A n-round, two-party protocol $\Pi = (P_0, P_1)$ to compute functionality f is said to be a strongly fair covert protocol for the bidirectional channel $\mathcal{B}$ if the following conditions hold:

- (External covertness): For any input $\bar{x}$, $\langle P_0(x_0), P_1(x_1)\rangle \approx \mathcal{B}$.
- (Strong Internal Covertness): There exists a PPT E (an *extractor*) such that if PPT $D(V)$ distinguishes between $V^{P_\sigma}_{\Pi,i}(\bar{x})$ and $V^{P_\sigma}_{\Pi:\mathcal{B}_{1-\sigma},i}(\bar{x})$ with advantage ϵ, $E^D(V^{P_\sigma}_{\Pi}(\bar{x}))$ computes $f(\bar{x})$ with probability at least $\epsilon/poly(k)$
- (Strong Fairness): If the functionality f is fair, then for any C_σ running in time T such that $\Pr[C_\sigma(V^{\sigma}_{\Pi,i}(\bar{x})) = f(\bar{x})] \geq \epsilon$, there exists a $C_{1-\sigma}$ running in time $O(T)$ such that $\Pr[C_{1-\sigma}(V^{1-\sigma}_{\Pi,i}(\bar{x})) = f(\bar{x})] = \Omega(\epsilon)$.
- (Final Covertness): For every PPT D there exists a PPT D' and a negligible ν such that for any x_σ and distribution $X_{1-\sigma}$, $\mathbf{Adv}^{D}_{V^{P_\sigma}_{\Pi}(X_{1-\sigma},x_\sigma),V^{P_\sigma}_{\Pi:\mathcal{B}_{1-\sigma}}(X_{1-\sigma},x_\sigma)}(k) \leq \mathbf{Adv}^{D'}_{f(X_{1-\sigma},x_\sigma),U_l}(k) + \nu(k)$.

Intuitively, the Internal Covertness requirement states that "Alice can't tell if Bob is running the protocol until she gets the answer," while Strong Fairness requires that "Alice can't get the answer unless Bob can." Combined, these requirements imply that neither party has an advantage over the other in predicting whether the other is running the protocol.

4.2 Construction

As before, we have two parties, P_0 (Alice) and P_1 (Bob), with inputs x_0 and x_1, respectively, and the function Alice and Bob wish to compute is $f : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \{0,1\}^{l}$, presented by the circuit C_f. The protocol proceeds in $2k+2$ rounds, where each round is a complete execution of the COVERT-YAO protocol presented in Section 3. At each round, one party creates a circuit which checks to see if the other party has obeyed the protocol. The difficulty to overcome is that the result of the check cannot be returned to the party creating the circuit without giving away that the other party is running the protocol. So instead, the parties take turns giving away one bit of information each round, and each party's circuit uses as input some pseudorandom "state" which was output by the last circuit this party created, carrying forward the information that the evaluator has not cheated. At the conclusion, each party can use the slowly revealed bits to compute $f(x_0, x_1)$. Let the function $G : \{0,1\}^k \rightarrow \{0,1\}^l$ be modeled by a random oracle, and let $F_K : \{0,1\}^k \rightarrow \{0,1\}^k{}_{K\in\{0,1\}^k}$ be a pseudorandom function. We let $commit(t;\rho_t), verify(\kappa_t;t,\rho_t)$ denote the commitment and verification functions, respectively, of a pseudorandom noninteractive commitment scheme.

Protocol 4 *(Fair covert two-party computation)*

To begin, each party P_σ chooses random strings $r_\sigma, K_\sigma \leftarrow U_k$. P_σ's inputs to the protocol are then $x_\sigma, r_\sigma, K_\sigma$. Each party P_σ computes commitments $\kappa_x^\sigma = commit(x_\sigma; \rho_x^\sigma)$, $\kappa_r^\sigma = commit(r_\sigma; \rho_r^\sigma)$, and $\kappa_K^\sigma = commit(K_\sigma; \rho_K^\sigma)$ and sends these commitments to the other party. The computation then proceeds in $2k+1$ rounds, each computing COVERT-YAO. Denote by $\kappa_r^\sigma[j]$ the commitment to $r_\sigma[j]$, the j^{th} bit of r_σ. Each round reveals one bit of r_0 or r_1, which conceal the final result.

$C_0(x_0, \rho_x^0) =$
if $(verify(\kappa_x^0; x_0, \rho_x^0) = true)$
  then set $R^0 = G(r_1) \oplus f(x_0, x_1)$
else draw $R^0 \leftarrow U_l$
output R^0

$C_1^0(x_1, \rho_x^1, r_1, \rho_r^1) =$
Let $R = f(x_0, x_1)$
if $(verify(\kappa_x^1; x_1, \rho_x^1) = true$
and $verify(\kappa_r^1; r_1, \rho_r^1) = true$
and $G(r^1) \oplus R = R^0$)
  then set $R^1 = G(r_0) \oplus R$,
$S_1^1 = F_{K_0}(1)$,
$b_1[1] = r_0[1]$.
else draw $R^1 \| S_1^1 \| b_1[1] \leftarrow U_{l+k+1}$
output $R^1 \| S_1^1 \| b_1[1]$

$C_1^1(x_0, \rho_x^0, r_0, \rho_r^0) =$
Let $R = f(x_0, x_1)$
if $(verify(\kappa_x^0; x_0, \rho_x^0) = true$
and $verify(\kappa_r^0; r_0, \rho_r^0) = true)$
and $G(r^0) \oplus R = R^1$)
  then set $S_1^0 = F_{K_1}(1)$,
$b_0[1] = r_1[1]$
else draw $S_1^0 \| b_0[1] \leftarrow U_{k+1}$
output $S_1^0 \| b_0[1]$

$C_j^\sigma(K_{1-\sigma}, \rho_K^{1-\sigma}, \rho_r^{1-\sigma}[j-1]) =$
if $(verify(\kappa_r^{1-\sigma}[j-1]; b_\sigma[j-1], \rho_r^{1-\sigma}[j-1]) = true$
and $verify(\kappa_K^{1-\sigma}; K_{1-\sigma}, \rho_K^{1-\sigma}) = true$
and $F_{K_{1-\sigma}}(j-1) = S_{j-1}^\sigma)$
  then set $S_j^{1-\sigma} = F_{K_\sigma}(j)$,
$b_{1-\sigma}[j] = r_\sigma[j]$
else draw $S_j^{1-\sigma} \| b_{1-\sigma}[j] \leftarrow U_{k+1}$
output $S_j^{1-\sigma} \| b_{1-\sigma}[j]$

Fig. 1. The circuits C_0, C_1^0, C_1^1, C_j^σ.

0. Bob garbles the circuit C_0 shown in figure 1, which takes x_0, ρ_x^0 as input and outputs $G(r_0) \oplus f(x_0, x_1)$ if κ_x^0 is a commitment to x_0. Bob and Alice perform the COVERT-YAO protocol, giving Alice the result R^0. If the check fails, R^0 is a uniformly chosen string and has no information about $f(x_0, x_1)$.
1. Alice garbles the circuit C_1^0 shown in figure 1, which takes $x_1, \rho_x^1, r_1, \rho_r^1$ as inputs and checks that κ_r^1 is a commitment to r_1, κ_x^1 is a commitment to x_1, and $R^0 = G(r_1) \oplus f(x_0, x_1)$; and outputs $G(r^0) \oplus f(x_0, x_1)$, a pseudorandom state $F_{K_0}(1)$, and the bit $r_0[1]$. Alice and Bob perform the COVERT-YAO protocol, giving Bob the result $R^1 \| S_1^1 \| b_1[1]$. If any of the checks fail, Bob's result is a uniformly chosen string which has no information about $f(x_0, x_1)$, otherwise $b_1[1] = r_0[1]$.
2. Bob garbles the circuit C_1^1 shown in figure 1, which takes $x_0, \rho_x^0, r_0, \rho_r^0$ as inputs and checks that κ_r^0 is a commitment to r_0, κ_x^0 is a commitment to x_0, and $R^1 = G(r_0) \oplus f(x_0, x_1)$; and outputs a pseudorandom state $F_{K_1}(1)$ and the bit $r_1[1]$. Bob and Alice perform the COVERT-YAO protocol, giving Alice the result $S_1^0 \| b_0[1]$. (So if both parties are following the protocol, $b_0[1] = r_1[1]$.)

   ⋮

i. Alice garbles the circuit C_j^0 (where $i = 2j-1$) which takes $K_1, \rho_K^1, \rho_r^1[j-1]$ as inputs and checks that $\kappa_r^1[j-1]$ is a commitment to $b_0[j-1]$, κ_K^1 is a commitment to K_1, and $S_{j-1}^0 = F_{K_1}(j-1)$, and if so outputs $F_{K_0}(j) \| r_0[j]$. The COVERT-YAO protocol gives Bob the result $S_j^1 \| b_1[j]$.

After $2k+1$ such rounds, if Alice and Bob have been following the protocol, we have $b_1 = r_0$ and $b_0 = r_1$ and both parties can compute the result. The pseudorandom "states" are what allow Alice

and Bob to check that all previous outputs and key bits (bits of r_0 and r_1) sent by the other party have been correct, without ever receiving the results of the checks or revealing that the checks fail or succeed.

Theorem 4. *Construction 4 is a strongly fair covert protocol realizing the functionality f*

Proof. The correctness of the protocol follows by inspection. The two-party security follows by the security of Yao's protocol. Now suppose that some party, wlog Alice, cheats (by sending a circuit which computes an incorrect result) in round $i = 2j - 1$. Then, the key bit $r_1[j]$ and state Alice computes in round $i + 1$ will be randomized; and with overwhelming probability every subsequent result that Alice computes will be useless. Assuming Alice can distinguish $f(x_0, X_1)$ from uniform, she can still compute the result in at most 2^{k-j} time by exhaustive search over the remaining key bits. By successively guessing the round at which Alice began to cheat, Bob can compute the result in time at most 2^{k-j+2}. If Alice aborts at round i, Bob again can compute the result in time at most 2^{k-j+1}. If Bob cheats in round i by giving inconsistent inputs, with high probability all of his remaining outputs are randomized; thus cheating in this way gives him no advantage over aborting in round $i - 1$. Thus, the fairness property is satisfied.

Neither Alice nor Bob can distinguish anything in their view from uniformly chosen bits without querying G at the random string chosen by the other. So given a distinguisher D running in time $p(k)$ for $V_{\Pi,i}^{P_0}(\bar{x})$ with advantage ϵ, it is simple to write an extractor which runs D, recording its queries to G, picks one such query (say, q) uniformly, and outputs $G(q) \oplus R^0$. Since D can only have an advantage when it queries r_B, E will pick $q = r_1$ with probability at least $1/p(k)$ and in this case correctly outputs $f(x_0, x_1)$. Thus the Strong Internal Covertness property is satisfied.

Weakly fair covertness. We can achieve a slightly weaker version of covertness without using random oracles. Π is said to be a *weakly fair* covert protocol for the channel $\mathcal{B}$ if Π is externally covert, and has the property that if f is strongly fair, then for every distinguisher D_σ for $V_{\Pi,i}^{P_\sigma}(\bar{x})$ with significant advantage ϵ, there is a distinguisher $D_{1-\sigma}$ for $V_{\Pi,i}^{P_{1-\sigma}}(\bar{x})$ with advantage $\Omega(\epsilon)$. Thus in a weakly fair covert protocol, we do not guarantee that both parties get the result, only that if at some point in the protocol, one party can tell that the other is running the protocol with significant advantage, the same is true for the other party.

We note that in the above protocols, if the function G is assumed to be a pseudorandom generator (rather than a random oracle), then the resulting protocol exhibits weakly fair covertness. Suppose D_σ has significant advantage ϵ after round $i = 2j$, as in the hypothesis of weak covertness. Notice that given $r_{1-\sigma}[1 \ldots j-1], G(r_{1-\sigma}) \oplus f(\bar{x})$, the remainder of P_σ's view can be simulated efficiently. Then D_σ must be a distinguisher for $G(r)$ given the first $j - 1$ bits of r. But since f is strongly fair, $P_{1-\sigma}$ can apply D_σ to $G(r_\sigma) \oplus f(\bar{x})$ by guessing at most 1 bit of r_σ and simulating P_σ's view with his own inputs. Thus $P_{1-\sigma}$ has advantage at least $\epsilon/2 - \nu(k) = \Omega(\epsilon)$.

5 Conclusions and Future Work

We have presented protocols for covert two-party computation that combine steganography with cryptographic protocols whose messages are all indistinguishable from uniformly chosen random bits. Covert two-party computation can be applied whenever the use of ordinary two-party computation raises enough suspicion to defeat its intended purpose. Our protocols are secure assuming the decisional Diffie-Hellman problem is hard.

Our work leaves room for improvement and open problems. For example, can fair covert two-party computation secure against malicious adversaries be satisfied without random oracles? It seems at least plausible that constructions based on concrete assumptions such as the "knowledge-of-exponent" assumption or the "generalized BBS" assumption may allow construction of such protocols, yet the obvious applications always destroy the final covertness property. More interestingly, can the notion of covert two-party computation be extended in some natural way to multiple parties?

References

1. G. Aggarwal, N. Mishra and B. Pinkas. Secure computation of the k'th-ranked element To appear in *Advances in Cryptology – Proceedings of Eurocrypt '04*, 2004.
2. L. von Ahn and N. Hopper. Public-Key Steganography. To appear in *Advances in Cryptology – Proceedings of Eurocrypt '04*, 2004.
3. M. Backes and C. Cachin. Public-Key Steganography with Active Attacks. *IACR e-print archive report 2003/231*, 2003.
4. M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. *Advances in Cryptology – Proceedings of CRYPTO '89*, pages 547-557, 1990.
5. E. Brickell, D. Chaum, I. Damgärd, J. van de Graaf: Gradual and Verifiable Release of a Secret. *Advances in Cryptology – Proceedings of CRYPTO '87*, pages 156-166, 1987.
6. C. Cachin. An Information-Theoretic Model for Steganography. *Information Hiding, 2nd International Workshop*, pages 306-318, 1998.
7. Z. Galil, S. Haber, M. Yung. Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model. *Advances in Cryptology – Proceedings of CRYPTO '87*, pages 135-155, 1987.
8. O. Goldreich. Secure Multi-Party Computation. Unpublished Manuscript. *http://philby.ucsd.edu/books.html*, 1998.
9. O. Goldreich, S. Goldwasser and S. Micali. How to construct pseudorandom functions. *Journal of the ACM*, vol 33, 1998.
10. O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game. *Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218-229.
11. J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4), pages 1364-1396, 1999.
12. N. Hopper, J. Langford and L. Von Ahn. Provably Secure Steganography. *Advances in Cryptology – Proceedings of CRYPTO '02*, pages 77-92, 2002.
13. M. Naor. Bit Commitment Using Pseudorandomness. J. Cryptology 4(2): 151-158 (1991)
14. M. Naor, B. Pinkas and R. Sumner. Privacy Preserving Auctions and Mechanism Design. *Proceedings, 1999 ACM Conference on Electronic Commerce.*
15. B. Pinkas. Fair Secure Two-Party Computation. In: *Advances in Cryptology – Eurocrypt '03*, pp 87–105, 2003.
16. L. Reyzin and S. Russell. Simple Stateless Steganography. IACR e-print archive report 2003/093, 2003.
17. A. C. Yao. Protocols for Secure Computation. *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pages 160–164.
18. A. C. Yao. How to Generate and Exchange Secrets. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pages 162–167.