Secure Computation using Leaky Tokens

Manoj Prabhakaran[1], Amit Sahai[2], and Akshay Wadia[3]

[1] UIUC mmp@uiuc.edu
[2] UCLA sahai@cs.ucla.edu
[3] UCLA awadia@cs.ucla.edu

Abstract. Leakage-proof hardware tokens have been used to achieve a large number of cryptographic tasks recently. But in real life, due to various physical attacks, it is extremely difficult to construct hardware devices that are guaranteed to be leakage-proof. In this paper, we study the feasibility of general two-party computation using leaky hardware tokens.

Our main result is a completeness theorem that shows that *every* non-trivial leaky two-party functionality can be used for general secure computation. In fact, the protocol we construct is *non-interactive* and *unconditionally secure.* There are no restrictions on the leakage functions associated with the token, except that it does not render the tokens trivial, by revealing its entire secrets to the adversary.

1 Introduction

Hardware tokens have received considerable attention in recent years ([1,3,4,6, 7,9–11,14,17,18,20,22,24,29,33,34], to name a few). In earlier works, specific hardware devices were used to achieve specific cryptographic tasks. Later on, as the potential of hardware devices was better understood, general hardware tokens were used to achieve general tasks. For example, in [20], among other things, the authors show feasibility of an unconditional, non-interactive UC protocol for general functionalities, using stateful hardware tokens.

In all these works, the hardware-tokens used are considered to be *leakage-proof*: there is a function f associated with the token (programmed with an input from the party creating the token) such that a party who receives it can only access the input/output behaviour of $f(\cdot)$, and cannot learn anything else about the input already programmed into it.

Although theoretically clean, in practice it is difficult to construct hardware devices that are truly leakage-proof. In particular, once the hardware device is in the adversary's possession, it can be subjected to *physical* attacks, like measuring power consumption, that can reveal secret information. (See, for example, [5,28, 35] and references therein, for various physical attacks.) Given that hardware devices cannot be guaranteed to be leakage-proof in real life, the natural question that arises is *whether we can use such leaky tokens to achieve cryptographic tasks.*

In this paper, we study the feasibility of using such leaky tokens for constructing general secure two-party computation protocols. Our focus is on theoretical feasibility and not practical efficiency, and we consider information-theoretic security. We think of a leaky token as a hardware one-time implementation of a two-party functionality $f(\cdot, \cdot)$. The sender of the token chooses an input x and creates such a token. The receiver can query the token with input y, at which point the token outputs $f(x, y)$. To model leakage, the adversary is allowed to ask a (possibly randomized) "leakage query" $L \in \mathcal{L}$, where $\mathcal{L}$ is a class of allowable leakage functions. To this, the token responds with the leakage $L(x)$. Once the token is accessed (using a legitimate input or a leakage function), it loses all information about x.

We point out a few aspects of our leakage model. On the positive side, the leakage functions are unrestricted (computationally unbounded, randomized, communicating arbitrary amount of information), and we prove the *best possible result* in that, unless a leakage function that completely reveals the functionality of the token is available to the adversary (in which case the hardware could as well be replaced with plain messages), the tokens can be used for secure evaluation of any function. On the other hand, we model the leakage function as acting on a single token at a time. But we do allow the adversary to adaptively choose the leakage function for each token, based on the information gathered from previously accessed tokens. Also, our leakage is one-time. We leave it for future work to consider models of *multi-round leakage* and *joint leakage* which would allow the adversary to, among other things, leak correlated information from multiple tokens.

Our Results. We construct a *non-interactive, unconditionally secure* general two-party secure computation protocol in the leaky token hybrid model. Instead of using leaky tokens which implement specific functionalities, we prove the following *general completeness theorem:* every *non-trivial* leaky two-party functionality is complete for unconditionally secure two-party computation. A two-party functionality is called "trivial" if there is a leakage function in the allowable leakage class that exhausts all the entropy from a token that was initialized with a uniformly chosen sender's input. Otherwise, the functionality is called non-trivial.

Related Work. There has been a long line of work on leakage resilient primitives, for example [12, 23, 32] and several more recent works. We mention just a few of them which are more relevant to our setting. Hardware leakage has been theoretically modeled and studied in several works including [15, 21, 23]; these works consider a model with no protected parts (or tokens), but with significantly restricted leakage functions. The study of leakage resilience for interactive protocols was initiated in [13], which constructed secure multi-party computation protocols using leaky tokens, but relying on computational assumptions. The problem of tamperable tokens was raised in [8], who showed that certain token functions are naturally resistant to certain restricted classes of tampering, or can be encoded to become so; this is applicable only when the tampering or leakage function applied to the token is from a well-behaved class. Hardware tokens have also been used to construct One Time Programs (OTP). An OTP for a function f allows a party to evaluate f on a single input x chosen by the party dynamically. OTPs are typically implemented as a package containing some software along with some hardware tokens. OTPs were introduced in [18]. In [20], OTPs were used to construct an unconditionally secure non-interactive protocol for two-party computation.

On the question of completeness of functionalities, building on his earlier results [25, 26], in 2000, Kilian [27] presented an elegant protocol relying on Nash equilibrium, to show that any non-trivial asymmetric functionality is complete for security against malicious adversaries; later, [30,31] provide further completeness results. The protocols presented in these works are all interactive. A *non-interactive* completeness theorem for non-trivial functionalities was first shown in [2]. However, all off these results are in the non-leaky model. Our theorem can be seen as a generalization of the completeness of non-trivial asymmetric functionalities [2, 27] to the leaky setting.

1.1 Technical Overview

Our goal is to use leaky tokens to construct a non-interactive, unconditionally secure two-party protocol for general functionalities. In [20], the authors construct such a protocol in the One Time Memory (OTM) hybrid model, which was introduced in [18]. The OTM functionality is a two-party functionality which, on input a pair of bits (b_0, b_1) from the sender and a bit c from the receiver, returns b_c to the receiver. That is, an OTM behaves like an implementation of Oblivious Transfer (OT), but with the following crucial difference: in OT, after the receiver specifies its input bit and receives its output b_c, the sender gets an acknowledgment that the receiver has received its output. However, in OTM, no such acknowledgment is sent to the sender (for more on the consequences of this crucial difference, see [18] and Section 4.2 in [20]). In light of the construction given in [20], to achieve our goal, it is sufficient to realize the OTM functionality using leaky tokens.

We consider a token to be a one-time evaluable implementation of a deterministic, constant-sized function f of two variables, into which one variable has been programmed. A user can evaluate the token only once, on any input of its choice. But, we allow an adversarial user to adaptively specify leakage functions for each token (which can be randomized and computationally unbounded), in lieu of feeding it a valid input. We shall show that *any such token is complete for token-based non-interactive secure computation*, as long as none of the allowed leakage functions fully reveals the information programmed into the token. Here, completeness means that *any function* that takes inputs from two parties and provides output to only one of them (the receiver) can be securely implemented using a token-based non-interactive protocol, even if the receiver is malicious and can leak from the tokens, adaptively. (The sender could be malicious as well.)

Technically, the work most relevant to ours is a recent result of [2]. There, a similar result is shown when there is no leakage from the tokens; also, a *deterministic extraction* procedure given there turns out to be useful for us. But handling arbitrary leakage presents several new technical challenges.

As a simple example, consider the following kind of token. It implements 1-out-of-m string OTM: i.e., the receiver can obtain one of m strings that are programmed into the token. It also permits a simple leakage function, which allows a corrupt receiver to learn $m-1$ out of the m strings. If $m = 2$ this function is the same as 1-out-of-2 OTM. But, for $m \geq 3$, can such a token be used to implement (non-leaky) 1-out-of-2 bit OTM?

It can in fact be argued that such a functionality, when implemented by a trusted third party, *cannot* be used to obtain OTM, even using interactive protocols! Suppose there is such an OTM protocol. Since the sender should not learn the receiver's input, the receiver will be able to run two simultaneous executions of this protocol, with two different inputs, while producing the same view for the sender; if the two executions have two different inputs to a 1-out-of-m OTM session, the receiver will use the leakage facility to learn both the inputs (since it is allowed to learn $m-1 \geq 2$ inputs). This would enable the receiver to complete both executions and learn both inputs of the sender, contradicting the security of the protocol. *However*, the token model differs from the standard hybrid model in a crucial way. In the standard hybrid model, both the parties learn about when each session is initiated as well as when the outputs are delivered in each session. In contrast, in the token model, the sender does not learn at what point — and in particular, in what order — the various tokens are accessed by the receiver.

This leads us to the following simple protocol for OTM based on this token. The sender sends two tokens for 1-out-of-m string OTM. Each of the m strings in the first token, contains a random bit; in addition, the first of these m strings contains a pointer to (i.e., the index of) one of the m strings in the next token. Similarly, the second token contains m random bits, and in addition the first bit is bundled with a pointer to one of the m strings. The random bit in the first token that is pointed to by the second token is used to mask the sender's first input; similarly, the second input is masked by the random bit in the second token that is pointed to by the first token. The masked values are sent along with the tokens. An honest receiver who wants to access the first input would open the pointer stored in the second token first, and find out the position in the first token where the mask bit is. Note that the order in which the receiver opens the two tokens reveals its input, but this remains hidden from the sender.

Now, in this protocol, a malicious receiver needs to access (using a leakage function) one of the two tokens at first; in doing so, it can learn up to $m-1$ strings in that token. But with probability $1/m$, the one string that it leaves out is the one that is pointed to by the other token. Thus, with probability at least $1/m$, the receiver learns only one of the two inputs of the sender. This uncertainty can be amplified by using the XOR of many such bits as the mask, to obtain a statistically secure bit OTM protocol.

The above class of functions and leakage are quite well-behaved. But in general, the leakage can be randomized, and comes from an infinite class of possible functions. Our main contribution is to show that, in spite of this, any token with non-trivializing leakage can be used to implement the above well-behaved class of functions and leakage. Our protocol is oblivious to the class of leakage functions (but does depend on the function f).[4]

The basic idea behind our protocol for implementing (leaky) 1-out-of-m OTM is similar to (but more sophisticated than) that of the 1-out-of-2 OT protocol in [2]. As in the OTM protocol there, roughly, our goal is to force the adversary to suffer a small amount of uncertainty about one of the inputs of the sender, and then amplify this uncertainty using many executions combined using a *deterministic extraction* strategy, which takes a sum of (small degree) products. (Standard randomized extraction is not applicable in this setting since the adversary can see the seed before it accesses the tokens, and hence the seed is not independent of the entropy source.) *But an additional challenge in our setting is that all of the leakage functions available to the adversary are not available to a honest receiver.* We need to translate this arbitrary gap between the power of the adversary and the honest receiver to that between the two in the leaky 1-out-of-m OTM functionality.

Another technical difference between the setting of [2] and ours is the following: when there is no leakage present (as in [2]), the non-triviality requirement on f implies that there are two specific inputs for the receiver which are "undominated" such that a secure protocol can be designed by ignoring the other inputs for f. That is, the honest receivers can be required to feed only one of these two inputs to f. In contrast, when leakage functions (even deterministic ones) are present, this is no more the case. This is exemplified by the leaky 1-out-of-m OTM itself: if the honest receiver is restricted to using only $m-1$ of the possible m inputs, then a leakage function would let a malicious receiver learn all the relevant secrets from the tokens, and then run the protocol with different inputs. Thus, in our case it is important that the honest receiver uses its entire domain of inputs.

Our solution involves a collection of m maps from the range of the function f to $\mathbb{Z}_p$ (for an appropriately chosen p), where m is the number of inputs to f that an honest receiver has. The tokens from the sender are grouped into "bundles" of $2m$ tokens each, two for each map. Within a bundle, the output from a token is meant to be mapped to $\mathbb{Z}_p$ using the map associated with that token, and then multiplied together. (The deterministic extraction in our case involves adding together the products from each bundle, modulo p.)

[4] For setting *concrete* parameters, one would need a concrete bound on the value of the entropy parameter measuring the non-triviality of the function in the presence of leakage.

The non-triviality of the function given the leakage function family guarantees that for every admissible leakage function L, there is some y such that $H(f(X,y)|L(X))$ is lowerbounded by a constant. Suppose $\widehat{y}$ be the y in a bundle for which this uncertainty is accumulated the most. The maps above are chosen in such a way that a fraction of this uncertainty will be preserved in the product corresponding to $\widehat{y}$, while for every other y, at least with a constant probability, its corresponding product will become fixed to 0. Thus, even given the product corresponding to all other y's (which is fixed to 0 with some probability), there is some uncertainty in the product corresponding to $\widehat{y}$. Finally, for each y, the products from all the bundles are summed together to extract a mask corresponding to y. The above property of the maps ensures that, for some $\widehat{y}$, the extracted value is close to uniformly random, even conditioned on the extracted values corresponding to the other $m-1$ y's.

2 Preliminaries

Protocols and Security. We follow standard definitions of protocol execution and security as specified in [19] and [16]. We will use $\mathcal{F}_{\mathsf{OTM}}$ to denote the OTM functionality of [18, 20]. We will also use a generalization of OTM to the case of 1-out-of-m OTM, where the sender holds m inputs (instead of 2), out of which the receiver selects one. The reason we focus on the One Time Memory functionality is because of the following theorem:

Theorem 1 ([20],Theorem 13). *Let $f(x,y)$ be a non-reactive, sender-oblivious, polynomial-time computable two-party functionality. Then there exists an efficient, statistically UC-secure non-interactive protocol which realizes f in the $\mathcal{F}_{\mathsf{OTM}}$-hybrid model.*

Modeling Leakage. Let $f : \mathsf{X}\times\mathsf{Y} \to \mathsf{Z}$ be a constant-size deterministic function. Let $\mathcal{L}$ be a set of possibly randomized *leakage functions*. To model leakage, we define the *token functionality* $\mathcal{F}^{(f,\mathcal{L})}$. The functionality receives $x \in \mathsf{X}$ from the sender. An honest receiver sends $y \in \mathsf{Y}$ to $\mathcal{F}^{(f,\mathcal{L})}$ and obtains $f(x,y)$. A malicious receiver, on the other hand, may query $\mathcal{F}^{(f,\mathcal{L})}$ with a leakage function $L \in \mathcal{L}$ and obtain $L(x)$. If L is a randomized leakage function, $L(x)$ specifies a distribution, and the ideal functionality samples $w \leftarrow L(x)$ and returns it to the receiver. For conciseness of notation, for randomized leakage functions L, we will use $L(x)$ to also mean a sample from that distribution.

Functionality $\mathcal{F}^{(f,\mathcal{L})}$:

- On receiving the message $(\mathsf{input}, \mathsf{sid}, P_j, x)$ from party P_i, store the tuple $(P_i, P_j, \mathsf{sid}, x)$ and send $(\mathsf{received}, P_i, \mathsf{sid})$ to party P_j. Ignore all input messages from P_i with session id sid.
- **Honest Query.** On receiving $(\mathsf{output}, \mathsf{sid}, P_i, y)$ from party P_j, if no tuple of the form $(P_i, P_j, \mathsf{sid}, x)$ exists, do nothing. Else, send $(\mathsf{output}, \mathsf{sid}, P_i, f(x,y))$ to party P_j, and delete the tuple $(P_i, P_j, \mathsf{sid}, x)$.

– **Leaky Query.** On receiving $(\mathsf{leak}, \mathsf{sid}, P_i, L)$ from party P_j, if $L \notin \mathcal{L}$ or no tuple of the form $(P_i, P_j, \mathsf{sid}, x)$ exists, do nothing. Else, send $(\mathsf{leak}, \mathsf{sid}, P_i, L(x))$ to party P_j, and delete the tuple $(P_i, P_j, \mathsf{sid}, x)$.

It is clear that no security is possible if the adversary is allowed arbitrary leakage queries. For example, if the adversary is allowed to ask the identity map as its leakage query, then it learns x and renders the functionality 'trivial'. To avoid this, we restrict the adversary to use only those leakage queries that leave some uncertainty *in the output* corresponding to some Bob's input $y \in \mathsf{Y}$. This is formalized below.

Definition 1. *Let $f : \mathsf{X} \times \mathsf{Y} \to \mathsf{Z}$ be a constant-size deterministic function, and let $\mathcal{L}$ be a set of functions with domain X. Let X be the uniform distribution on X. Then, the token functionality $\mathcal{F}^{(f,\mathcal{L})}$ is called* non-trivial *if there exists a constant $c > 0$ such that*

$$\min_{L \in \mathcal{L}} \max_{y \in \mathsf{Y}} H(f(X, y) \mid L(X)) \geq c.$$

Amplifying Uncertainty. In our proofs, we will consider random variables that have constant uncertainty conditioned on the adversary's view. To amplify this uncertainty, we will take the sum of an appropriate number of such random variables. To this end, the following generalization of a lemma from [2] will be useful.

Lemma 1. *Let p be a fixed prime number. For any positive integer N, let $X_1, \cdots, X_N$ be N independent random variables over the alphabet $\mathbb{Z}_p$, such that for some constant $\epsilon > 0$, for all $i \in [N]$, for all $z \in \mathbb{Z}_p^N$, $\Pr[X_i = z] < 1 - \epsilon$. Then the statistical distance between the distribution of $\sum_{i=1}^{N} X_i$ (summation in $\mathbb{Z}_p$) and the uniform distribution over $\mathbb{Z}_p$ is negligible in N.*

3 Leaky-OTM from Leaky Tokens

In this section we show how to use any non-trivial leaky token to implement a functionality $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$ described below. Here, the parameter m is equal to the number of inputs for (honest) Bob to the token. We leave out from the notation of $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$ a parameter d specifying the length of the "messages" that Alice gives as input to $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$. d can be set to any constant in our following construction, by choosing the parameter $p \geq 2^d$.

Functionality $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$:

– **When Bob is honest:** function as a 1-out-of-m (string) OTM. i.e., accept m strings, each d bits long, $(x_1, \cdots, x_m)$ from Alice, and an index $i \in [m]$ from Bob. Output x_i to Bob.

– **When Bob is corrupt:** function as $(m-1)$-out-of-m OTM. Here, Alice's input is the same as above; Bob can input any set $S \subsetneq [m]$ and receive $\{x_j | j \in S\}$ as output.

Let $\mathcal{F}$ be any non-trivial leaky token functionality, with m inputs for (honest) Bob. The idea behind our protocol for $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ is a generalization of an earlier protocol for OTM from non-trivial (non-leaky) tokens [2]. The main complication in allowing leakage is that honest Bob does not have access to all the inputs an adversarial Bob can have. In particular, unlike in [2], it is not true that we can find two "undominated" inputs for Bob. Indeed, restricting to any strict subset of the m inputs can render the function trivial in the presence of the leakage functions. This complicates our construction when $m > 2$. (When $m = 2$, one could use the protocol in [2], but our protocol does not become identical to that in [2] if we set $m = 2$.)

While the protocol in [2] considered tokens in bundles of two, we shall use bundles of size $2m$. Further, unlike in that protocol where the same set of maps were used in all tokens to map Bob's (input, output) pairs to elements in a field, we use m different sets of maps for the $2m$ tokens in the bundle, to map these pairs to $\mathbb{Z}_p$ (for a large enough prime constant p). We give the formal description of the protocol below, and defer the proof of security to Appendix B.

Set up: Let $\mathcal{F} = (f, \mathcal{L})$ be a non-trivial leaky functionality evaluating a function $f : \mathsf{X} \times \mathsf{Y} \to \mathbb{Z}_p$ for some prime p[5] and allowing a set of leakage functions $\mathcal{L}$ such that for all $L \in \mathcal{L}$, $\min_{y \in \mathsf{Y}} H(f(X, y)|L(X)) > c$ for some constant c, where X is uniformly distributed over X.

Alice can send tokens for $\mathcal{F}$ with an input $x \in \mathsf{X}$ of her choice. An honest Bob can evaluate the token once with $y \in \mathsf{Y}$ of his choice to obtain $f(x, y)$. Let $m = |Y|$ be the number of inputs for (honest) Bob. An adversarial Bob can evaluate $L(x)$ for a leakage function $L \in \mathcal{L}$ of his choice.

Output Maps: Define m maps $M_1, \cdots, M_m$, of the form $M_i : \mathsf{Y} \times \mathbb{Z}_p \to \mathbb{Z}_p$ as follows. Fix an arbitrary input $x^* \in \mathsf{X}$, and let $z_j^* = f(x^*, y_j)$. For $i = 1$ to m and $j = 1$ to m, define

$$M_i(y_j, z) = \begin{cases} z - z_j^* + 1 & \text{if } j = i, \\ z - z_j^* & \text{if } j \neq i \end{cases}$$

This ensures that for each i, j, $M_i(y_j, \cdot)$ is a permutation over $\mathbb{Z}_p$, and for each y_i, there is a map (namely M_i) such that $M_i(y_i, f(x^*, y_i)) = 1$ but $M_i(y_j, f(x^*, y_j)) = 0$ for all $j \neq i$.

Alice's program: Alice's input is m elements in $\mathbb{Z}_p$, $(s_1, \cdots, s_m)$.

1. Alice carries out the following computations:

[5] W.l.o.g, the output alphabet can be considered $\mathbb{Z}_p$ by choosing a prime p such that for each $y \in \mathsf{Y}$, $|\{f(x, y)|x \in \mathsf{X}\}| \leq p$.

- For $\ell = 1$ to κ, for $i = 1$ to m and $t \in \{1, 2\}$:
  - Pick $x_{\ell,i,t} \leftarrow \mathsf{X}$.
  - For $j = 1$ to m, let $R^j_{\ell,i,t} = M_i(y_j, f(x_{\ell,i,t}, y_j))$.
- For $j = 1$ to m:
  - Let $\overline{R}^j_\ell = \Pi^m_{i=1}\Pi^2_{t=1}R^j_{\ell,i,t}$.
  - let $r_j = \sum^\kappa_{\ell=1}\overline{R}^j_\ell$.
  - For $j = 1$ to m, let $z_j = s_j + r_j$.

2. Alice creates several $\mathcal{F}$-tokens with the following inputs:
   - Tokens labeled (ℓ, i, t) for $\ell \in [\kappa]$, $i \in [m]$, $t \in \{1, 2\}$, with inputs $x_{\ell,i,t}$.
   - $m\lceil \log p\rceil$ more tokens to "communicate" the bits of $(z_1, \cdots, z_m)$: in a token to send a bit 0, use input $\hat{x}^0$, and in a token to send a bit 1, use input $\hat{x}^1$, where $\hat{x}^0$ and $\hat{x}^1$ are such that $f(\hat{x}^0, y^*) \neq f(\hat{x}^1, y^*)$ for some y^*.

Bob's program: Bob's input is an index $c \in [m]$.

1. Bob accesses the $\mathcal{F}$-tokens sent by Alice with the following inputs:
   - For the tokens labeled (ℓ, i, t) for $\ell \in [\kappa]$, $i \in [m]$, $t \in \{1, 2\}$, use input y_c to obtain an output $z_{\ell,i,t} = f(x_{\ell,i,t}, y_c)$.
   - Use input y^* in the remaining tokens to learn $(z_1, \cdots, z_m)$ (or just z_c).
2. For all (ℓ, i, t) let $R^c_{\ell,i,t} = M_i(y_c, z_{\ell,i,t})$; let $\overline{R}^c_\ell = \Pi^m_{i=1}\Pi^2_{t=1}R^c_{\ell,i,t}$.
3. Let $r_c = \sum^\kappa_{\ell=1}\overline{R}^c_\ell$. Then output $s_c = z_c - r_c$.

---

4 OTM from Leaky-OTM

Suppose Alice and Bob parties have access to parallel copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$, for some constant m. If $m = 2$, $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ is the same as OTM. In this section we show how to implement OTM non-interactively, using parallel copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ and a single message from Alice to Bob, even when $m \geq 3$. The protocol uses the deterministic extraction strategy of [2]. The protocol crucially relies on the fact that Alice does not learn when Bob accesses various copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ (which are implemented using tokens). Correspondingly, Alice never learns when Bob accesses the OTM that is being implemented.

Overview. Firstly, note that implementing OTM using $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ tokens (for $m \geq 3$) is in fact impossible using a non-interactive protocol in which the order in which Bob accesses the token is *not adaptive*: Bob could mentally run two executions with two different values for his choice bits. Each execution would require the honest Bob to access at most one out of m positions in each $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ instance. But an adversarial Bob can access $m - 1 \geq 2$ positions in each instance. Thus he can complete both executions successfully, and learn both inputs of Alice. (A similar argument can be used to rule out implementing OTM in $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$-hybrid is used where Alice learns when Bob accesses an $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ instance, even if interaction is allowed, as long as Bob is computationally unbounded.)

So necessarily we shall need to rely on Bob being able to access the tokens adaptively, and in different order. This leads us to the following basic idea underlying our construction. Alice's inputs are hidden in two tokens, at random positions. In addition, the first token (in a fixed position) contains a pointer to the position in the second token where an input is hidden; similarly, the second token (in a fixed position) has a pointer to the position in the first token that holds the other input. To access the first input, Bob first opens the second token at the fixed position, to recover a pointer to the first token, and then accesses the first token to recover the input stored in that position from the first token. Similarly, if Bob wants to access the second input, he should first open the first token, recover a pointer that tells him which position in the second token he should access, and then open the second token at that position.

A malicious Bob must first open one of the two tokens. When he opens one token, and recovers $m-1$ positions, there is a $1/m$ probability that the one position that he did not access is the one containing Alice's input in that token. This gives us a weak form of OTM, in which, with some probability, Bob learns at most one secret.

To amplify this to a full-fledged OTM using a non-interactive protocol, several individual secrets are additively combined to form a random mask that is then used to mask the actual input of Alice. We give the formal description of the protocol below, and defer the proof of security to Appendix C.

Alice's program: Alice's input is two bits s_0, s_1.

1. Alice carries out the following computations:
   - For $i = 1$ to κ,
     - pick $x_i^0, x_i^1 \leftarrow [m]$ and for each $j = 1$ to m, pick $b_{i,j}^0, b_{i,j}^1 \leftarrow \{0,1\}$.
     - let $a_{i,1}^0 = (x_i^0, b_{i,1}^0)$ and $a_{i,1}^1 = (x_i^1, b_{i,1}^1)$; for $j = 2$ to m, let $a_{i,j}^0 = (0, b_{i,j}^0)$ and $a_{i,1}^1 = (0, b_{i,j}^1)$; for $j = 2$ to m.
     - let $R_i^0 = b_{i,x_i^1}^0$ and $R_i^1 = b_{i,x_i^0}^1$.
   - Let $r_0 = \sum_{i=1}^{\kappa} R_i^0$ and $r_1 = \sum_{i=1}^{\kappa} R_i^1$ (summation in $\mathbb{Z}_2$).
   - Let $z_0 = s_0 + r_0$ and $z_1 = s_1 + r_1$.
2. Alice invokes, in parallel, several copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ with the following inputs:
   - Sessions labeled $(i, \beta) \in [\kappa] \times \{0,1\}$ with input $(a_{i,1}^\beta, \ldots, a_{i,m}^\beta)$.
   - Another session of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ to "communicate" the bits (z_0, z_1): Alice can use $(z_0, z_1, 0, \cdots, 0)$ as her input in this session. (An adversary may receive both these bits, if $m \geq 3$.)

Bob's program: Bob's input is a choice bit b.

1. Bob invokes the same copies of $\mathcal{F}$ as Alice with the following inputs:
   – For $i = 1$ to κ,
     - If $b = 0$, first access the session numbered $(i, 1)$ with input 1, and obtain x_i^1, and then access the session numbered $(i, 0)$ with input x_i^1 to obtain the bit R_i^0.

  - If $b = 1$, first access session $(i, 0)$ with input 1, recover x_i^0 and then access the session $(i, 1)$ with input x_i^0 to recover R_i^1.
- Recover z_b from the last session of $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$, using b as input.

2. After all sessions of $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$ are completed, compute $r_b = \sum_{i=1}^{\kappa} R_i^b$, and $s_b = z_b - r_b$ (all operations in $\mathbb{Z}_2$). Output s_b.

References

1. Shashank Agrawal, Prabhanjan Ananth, Vipul Goyal, Manoj Prabhakaran, and Alon Rosen. Lower bounds in the hardware token model. To appear in TCC, 2014.
2. Shweta Agrawal, Vipul Goyal, Abhishek Jain, Manoj Prabhakaran, and Amit Sahai. New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In *CRYPTO*, pages 443–460, 2012.
3. Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N Rothblum. Program obfuscation with leaky hardware. In *Advances in Cryptology–ASIACRYPT 2011*, pages 722–739. Springer, 2011.
4. Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *CRYPTO*, pages 302–318, 1993.
5. David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.
6. Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for uc secure computation using tamper-proof hardware. In *EUROCRYPT*, pages 545–562, 2008.
7. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.
8. Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: built-in tamper resilience. In *Advances in Cryptology–ASIACRYPT 2011*, pages 740–758. Springer, 2011.
9. Ronald Cramer and Torben P. Pedersen. Improved privacy in wallets with observers (extended abstract). In *EUROCRYPT*, pages 329–343, 1993.
10. Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Isolated proofs of knowledge and isolated zero knowledge. In *EUROCRYPT*, pages 509–526, 2008.
11. Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *Theory of Cryptography*, pages 164–181. Springer, 2011.
12. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 293–302. IEEE, 2008.
13. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *Advances in Cryptology–CRYPTO 2011*, pages 297–315. Springer, 2011.
14. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.
15. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.
16. Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

17. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
18. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.
19. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
20. Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. *IACR Cryptology ePrint Archive*, 2010:153, 2010.
21. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.
22. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
23. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
24. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007.
25. Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
26. Joe Kilian. A general completeness theorem for two-party games. In *STOC*, pages 553–560, 1991.
27. Joe Kilian. More general completeness theorems for secure two-party computation. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 316–324. ACM, 2000.
28. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in CryptologyCRYPTO99*, pages 388–397. Springer, 1999.
29. Vladimir Kolesnikov. Truly efficient string oblivious transfer using resettable tamper-proof tokens. In *Theory of Cryptography*, pages 327–342. Springer, 2010.
30. Daniel Kraschewski, Hemanta K. Maji, Manoj Prabhakaran, and Amit Sahai. A full characterization of completeness for two-party randomized function evaluation. To appear in Eurocrypt, 2014.
31. Daniel Kraschewski and Jörn Müller-Quade. Completeness theorems with constructive proofs for finite deterministic 2-party functions. In *Theory of Cryptography*, pages 364–381. Springer, 2011.
32. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
33. Tal Moran and Moni Naor. Basing cryptographic protocols on tamper-evident seals. *Theor. Comput. Sci.*, 411(10), 2010.
34. Tal Moran and Gil Segev. David and goliath commitments: Uc computation for asymmetric parties using tamper-proof hardware. In *EUROCRYPT*, pages 527–544, 2008.
35. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.

A Protocols and Security

Following in [19] and [16], a protocol is represented as a system of probabilistic interactive Turing machines (ITMs), where each ITM represents the program to be run within a different party. Specifically, the input and output tapes model inputs and outputs that are received from and given to other programs running on the same machine, and the communication tapes model messages sent to and received from the network. Adversarial entities are also modeled as ITMs.

To argue security of a protocol, we proceed as follows: first, an *ideal functionality* is defined, which is a "trusted party" that is guaranteed to accurately capture the desired functionality. Then, the process of executing a protocol in the presence of an adversary is formalized. This is called the *real-life* model. Finally, an *ideal process* is considered, where the parties only interact with the ideal functionality, and not amongst themselves. Informally, a protocol realizes an ideal functionality if running of the protocol amounts to "emulating" the ideal process for that functionality

Let $\Pi = (P_1, P_2)$ be a protocol, and $\mathcal{F}$ be the ideal-functionality. We describe the ideal and real world executions. The *real-life model* consists of the two parties P_1 and P_2, and the adversary A. Adversary A can corrupt any party. When A corrupts party P_i , it learns P_i's entire internal state, and takes complete control of P_i's input/output behaviour. Let $real_{\Pi,A}(\kappa, x, y)$ be the distribution that describes the joint distribution of the outputs of the honest party and the adversary when protocol Π is run with security parameter κ, inputs x and y for P_1 and P_2 respectively, and adversary A.

The *ideal process* consists of two "dummy parties" $\hat{P}_1$ and $\hat{P}_2$, the ideal functionality $\mathcal{F}$, and the ideal world adversary S, called the *simulator*. In the ideal world, the uncorrupted dummy parties simply hand over their inputs to $\mathcal{F}$. As in the real world, adversary S can corrupt any party. Once it corrupts party $\hat{P}_i$, it learns $\hat{P}_i$'s input, and takes complete control of its input/output behaviour. Let $ideal_{\mathcal{F},S}(\kappa, x, y)$ be the random variable describing the joint distribution of the outputs of the honest party and the adversary in the ideal world.

We say that a protocol Π *securely realizes* functionality $\mathcal{F}$ if for every real world adversary A, there exists a (possibly unbounded) ideal world simulator S, such that for all inputs x and y, the ensembles $\{\, real_{\Pi,A}(\kappa, x, y) \,\}_\kappa$ and $\{\, ideal_{\mathcal{F},S}(\kappa, x, y) \,\}_\kappa$ are statistically close.

B Proof of Security of the Leaky-OTM Protocol

B.1 Proof of Lemma 1

In this section, present the proof of Lemma 1. The proof is along the lines of the proof for a simpler for this lemma, from [2]. First, we prove a special case of Lemma 1, obtained by restricting the support of all random variables to a set of two elements.

Lemma 2. *Let p be a fixed prime number. For any positive integer N, let $X_1, \cdots, X_N$ be independent random variables over the alphabet $\mathbb{Z}_p$, such that there exist $\alpha, \beta \in \mathbb{Z}_p$ and some constant $\epsilon > 0$, such that for all $i \in [N]$, the* *support of X_i is $\{\alpha, \beta\}$, and for all $i \in [N]$, and $\epsilon \leq \Pr[X_i = \alpha] \leq 1 - \epsilon$. Then the statistical distance between the distribution of $\sum_{i=1}^{N} X_i$ (summation in $\mathbb{Z}_p$) and the uniform distribution over $\mathbb{Z}_p$ is negligible in N.*

Proof. Firstly, define $X_i' = X_i - \beta$, and $z = \alpha - \beta$, so the support of X_i' is $\{0, z\}$. Also, let $\delta_i := \Pr[X_i' = z]$. For $w \in \mathbb{Z}_p$, we analyze $\Pr\left[\sum_i X_i' = w\right]$. Let $u = wz^{-1}$ (where $u \in \mathbb{Z}_p$ is treated as an integer in $\{0, \cdots, p-1\}$). Let $S_k(\xi)$ be the formal polynomial in ξ defined as

$$S(\xi) = \Pi_{i=1}^{N}((1 - \delta_i) + \delta_i \xi).$$

Let C_d be the coefficient of ξ^d in $S(\xi)$. Then $\Pr\left[\sum_i X_i' = w\right] = \sum_{t \geq 0} C_{tp+u}$. But we have $\sum_{t \geq 0} C_{tp+u} = \frac{1}{p} \sum_{k=0}^{p-1} \omega^{-ku} S(\omega^k)$, where $\omega = e^{i2\pi/p}$, because $\omega^{-ku} \cdot (\omega^k)^{tp+u} = 1$ and $\sum_{k=0}^{p-1} \omega^{kr} = 0$ for all $r \in \mathbb{Z}_p \setminus \{0\}$. Hence,[6]

$$\begin{aligned}
\Pr\left[\sum_i X_i' = w\right] &= \sum_{t \geq 0} C_{tp+u} = \frac{1}{p} \sum_{k=0}^{p-1} \omega^{-ku} S(\omega^k) \\
&= \frac{1}{p} + \frac{1}{p} \sum_{k=1}^{p-1} \omega^{-ku} S(\omega^k) && \text{because } S(1) = 1 \\
&= \frac{1}{p} \pm \frac{1}{p} \sum_{k=1}^{p-1} |S(\omega^k)| \\
&= \frac{1}{p} \pm \frac{1}{p} \sum_{k=1}^{p-1} (1 - \alpha)^N && \text{where } \alpha := 1 - |(1 - \epsilon) + \epsilon\omega| \\
&= \frac{1}{p} \pm (1 - \alpha)^N
\end{aligned}$$

The bound on $|S(\omega^k)| \leq (1 - \alpha)^N$ used above holds because, for $1 \leq k \leq p - 1$, for all $i \in [N]$, $|(1 - \delta_i) + \delta_i \omega^k| \leq 1 - \alpha$, since for each i, $\delta_i \in [\epsilon, 1 - \epsilon]$. Note that α is a constant (since p and ϵ are constants independent of N) such that $0 < \alpha < 1$, and hence $(1 - \alpha)^N$ is negligible in N. This concludes the proof that $\sum_i X_i'$, and hence $\sum_i X_i$, are distributed almost uniformly over $\mathbb{Z}_p$.

Proof of Lemma 1. Now we complete the proof of Lemma 1, using Lemma 2. Here, the random variables X_i can all have different supports, but each support should be of size at least two. With each X_i, we associate a pair (z_0^i, z_1^i) as the two values in $\mathbb{Z}_p$ which X_i has the maximum probability of taking (breaking ties arbitrarily). Note that X_i should have a mass of at least ϵ/p on each of z_0^i, z_1^i. Since there are only a constant number (i.e., $\binom{p}{2}$) possible values for the pairs (z_0^i, z_1^i), we can fix a pair α, β such that at least a constant fraction of the N X_i's have $(z_0^i, z_1^i) = (\alpha, \beta)$. Let the random variable $T = \{i | (z_0^i, z_1^i) =$

[6] The notation $a = b \pm c = b \pm d$ stands for $a \in [b - c, b + c] \subseteq [b - d, b + d]$.

(α, β) and $X_i \in \{\alpha, \beta\}\}$. There is a constant γ such that with all but negligible probability $|T| \geq \gamma N$. Conditioned on this, for each setting of T, we can write $\sum_{i=1}^{N} X_i = Y_0 + \sum_{i=1}^{\gamma N} Y_i$, where the Y_i are independent of each other, and Y_i, for $i > 0$, satisfy the condition of Lemma 2. Then it follows from Lemma 2 that $\sum_{i=1}^{\gamma N} Y_i$ is almost uniformly ditributed over $\mathbb{Z}_p$, and hence so is $\sum_{i=1}^{N} X_i$. Since this holds for every value of T (when $|T| \geq \gamma N$), we conclude that $\sum_{i=1}^{N} X_i$ is almost uniformly ditributed over $\mathbb{Z}_p$.

B.2 Simulation and Proof

Description of the Simulator. To construct the simulator, we follow the approach of [2] and divide the invocation of $\mathcal{F}^{(f,\mathcal{L})}$ into the following imaginary phases:

- Communication Phase: W.l.o.g., we assume that Bob first schedules the communication sessions, and uses the input y^* prescribed for the communication sessions. (Any other strategy is "dominated" by this, in that it can be simulated by an adversary as above.)
- Phase 1: This phase starts when Bob invokes one of the $2\kappa m$ sessions, and lasts until, for $c_1\kappa$ values of $\ell \in [\kappa]$, at least one $x_{\ell,i,t}$ has been invoked. Here, c_1 is a constant to be determined later.
- Phase 2: This phase consists of the remaining invocations of $\mathcal{F}^{(f,\mathcal{L})}$.

When the adversary queries the token $x_{\ell,i,t}$ with leakage function L, some constant entropy is left in one of the columns, i.e., there exists $j \in [m]$ such that $H(f(x_{\ell,i,t}, y_j) \mid L(x_{\ell,i,t})) > c$. We show later that with constant probability, this entropy is translated into uncertainty about $\overline{R}_\ell^j$. To formalize this notion, we have the following definition.

Definition 2. *For $\ell \in [\kappa]$ and $j \in [m]$, let (x, t) be the first session invoked by the adversary in bundle ℓ. Further, let L be the leakage query. We say that "ℓ is potentially-j-undetermined" if $H(f(x_{\ell,i,t}, y_j) \mid L(x_{\ell,i,t})) > c$.*

As the functionality is not trivial, each token invoked by the adversary leaves some entropy in one of the columns. Thus, for each bundle ℓ that is invoked (that is, there exists some token $x_{\ell,i,t}$ in bundle ℓ which has been invoked by the adversary), there is some $j \in [m]$ such that ℓ is potentially-j-undetermined. We now describe the simulator.

- Communication Phase: Uniformly pick $\hat{s}_1, \ldots, \hat{s}_m \leftarrow \mathbb{Z}_p$, and faithfully simulate all communication sessions using these as the sender's inputs.
- Phase 1: To simulate a new session (ℓ, i, t), accept $y_{\ell,i,t}$ from Bob and return $f(x_{\ell,i,t}, y_{\ell,i,t})$ for a uniformly chosen input $x_{\ell,i,t}$ in X.
- Input Extraction: At the end of Phase 1, there exists $\widehat{\jmath} \in [m]$, such that at least $c_1\kappa/m$ bundles invoked by the adversary in Phase 1 are potentially-$\widehat{\jmath}$-undetermined. The simulator sends $\mathbf{j} = [\kappa]\backslash\{\widehat{\jmath}\}$ to $\widetilde{\mathcal{F}}_{\mathsf{OTM}}^{(m)}$ and obtains $(m-1)$ messages $(s_1, \ldots, s_{\widehat{\jmath}-1}, s_{\widehat{\jmath}+1}, \ldots, s_m)$.

– Phase 2: The simulator answers the remaining queries by selecting $x_{\ell,i,t}$ uniformly at random, conditioned on $r_j = s_j - \hat{s}_j$ for $j \neq \widehat{\jmath}$.}

The correctness of simulation rests on the following claims about the real execution.

Claim. In the real execution, the sequence $(r_1, \ldots, r_m)$ is almost (i.e., up to negligible statistical distance) uniformly distributed over $\mathbb{Z}_p^m$.

Proof. We will show that for each $j \in [m]$, there exist a large number of indices $\ell \in [\kappa]$ such that $\overline{R}_\ell^j$ has constant entropy, even conditioned on $\overline{R}_\ell^{j'}$ for $j' \neq j$. For a fixed $\ell \in [\kappa]$ and $j \in [m]$, we say "ℓ is j-good" if the following event occurs:

- $x_{\ell,j,0} = x^*$ and,
- $M_i(y_j, f(x_{\ell,i,t}, y_j)) \neq 0$ for $i \neq j$ and $t \in \{0,1\}$.

First note that the above event occurs with constant probability. This is because as the functionality $f(\cdot,\cdot)$ is not trivial, each column contains at least two distinct values. Further, as the map $M_i(y_j,\cdot)$ is one-to-one, for at least one of these distinct values, we have $M_i(y_j, f(x_{\ell,i,t}, y_j)) \neq 0$. Now, conditioned on the above event, we have that

- $\overline{R}_\ell^{j'} = 0$ for $j' \neq j$. This is because $x_{\ell,j,0} = x^*$, and therefore, $M_j(y_{j'}, f(x^*, y_{j'})) = 0$.
- $\overline{R}_\ell^j = z \cdot M_j(y_j, f(x_{\ell,j,1}, y_j))$, for some $z \neq 0$.

Thus, when ℓ is j-good, $\overline{R}_\ell^j$ has at least constant entropy, even given $\overline{R}_\ell^{j'}$ for $j' \neq j$ (as $\overline{R}_\ell^{j'} = 0$ independently of $\overline{R}_\ell^j$). Now, let S_j be the set of indices which are j-good, i.e., $S_j := \{\, \ell \in [\kappa] \mid \ell \text{ is } j\text{-good}\}$. Then, with all but negligible probability, $|S_j| = \Omega(\kappa)$. We can write r_j as $r_j = u_j + v_j$, where $u_j := \sum_{\ell \in S_j} \overline{R}_\ell^j$. To show that the r_j's are (almost) uniformly distributed, it is sufficient to show that jointly the u_j's are (almost) uniformly distributed, given the v_j's. In fact, even given $\{\overline{R}_\ell^j | \ell \notin S_j\}$ (which determines v_j) for all j, each element in $\{\overline{R}_\ell^j | \ell \in S_j\}$ has at least a constant amount of entropy. The claim now follows directly from Lemma 1.

Claim. In the real execution, conditioned on a sequence $(r_1, \ldots, r_m)$, the $x_{\ell,i,t}$ invoked by the adversary in Phase 1 are almost uniformly distributed.

Proof. To show this, we show that the random variables $x_{\ell,i,t}$ invoked in Phase 1 are independent of $(r_1, \ldots, r_m)$. Let $T \subset [\kappa]$ be the subset of indices ℓ such that after Phase 1, none of the m tokens $x_{\ell,i,t}$ for $i \in [m]$ have been invoked. Note that $|T| = (1 - c_1)\kappa$. For each $j \in [m]$, let $S_j \subset T$ be the subset of indices that are j-good. Then, as before, we can write $r_j = u_j + v_j$. As before, the u_js are independent and (almost) uniformly distributed, and thus r_j is independent of all $x_{\ell,i,t}$ invoked in Phase 1.

Claim. Let $\widehat{\jmath}$ be defined for the real execution as it is defined in the simulation. Then $r_{\widehat{\jmath}}$ is almost uniformly distributed, even conditioned on $(r_1, \ldots, r_{\widehat{\jmath}-1}, r_{\widehat{\jmath}+1}, \ldots, r_m)$ and the adversary's view in Phases 1 and 2.

Proof. We will analyze queries the adversary makes in Phase 1, and show that with constant probability, each query leaves some uncertainty in some $\overline{R}^j_\ell$. This small amount of certainty can then be amplified using Lemma 1. Formally, consider an index $\ell \in [\kappa]$, and let x_{ℓ,i_0,t_0} be the first token with index ℓ invoked by the adversary in Phase 1, using some leakage function $L \in \mathcal{L}$. Since the functionality of the token $\mathcal{F}$ is non-trivial, we know that with some constant probability there is a $j \in [m]$ such that $L(x_{\ell,i_0,t_0})$ does not completely determine $f(x_{\ell,i_0,t_0}, y_j)$. Then, we shall argue, with constant probability, $x_{\ell,i,t}$ for all $(i,t) \neq (i_0, t_0)$ will be such that:

- $\overline{R}^{j'}_\ell = 0$ for all $j' \neq j$ (irrespective of x_{ℓ,i_0,t_0}), and
- $\overline{R}^j_\ell = z \cdot \pi(f(x_{\ell,i_0,t_0}, y_j))$, for $z \neq 0$, and π a permutation.

When the above event occurs, we will say that bundle ℓ is "j-undetermined". Then to complete the argument, we may consider giving the adversary $x_{\ell,i,t}$ for all $(i,t) \neq (i_0,t_0)$ (in addition to $L(x_{\ell,i_0,t_0})$, so that the adversary can compute z, π and $\overline{R}^{j'}_\ell = 0$ for all $j' \neq j$, and its uncertainty about $f(x_{\ell,i_0,t_0}, y_j)$ will translate to uncertainty about $\overline{R}^j_\ell$.

To see the first point above, note that in this bundle there is some *other* token labeled (ℓ, i_1, t_1) such that $i_1 = j$ (if $i_0 = j$ as well, we can take $t_1 = 1 - t_0$). With a constant probability (namely $1/|\mathsf{X}|$), $x_{\ell,i_1,t_1} = x^*$, so that for all $j' \neq j$ we have $R^{j'}_{\ell,i_1,t_1} = 0$ which implies $\overline{R}^{j'}_\ell = 0$ (independent of x_{ℓ,i_0,t_0}).

To see the second point, firstly note that if $x_{\ell,i_1,t_1} = x^*$, then $R^j_{\ell,i_1,t_1} \neq 0$ (in fact, 1) by choice of M_j. Further, with some constant probability, for all the tokens with label (ℓ, i, t) in this bundle such that $(i,t) \neq (i_0,t_0)$ and $(i,t) \neq (i_1,t_1)$, we will have $M_i(y_j, f(x_{\ell,i,t}, y_j)) \neq 0$. This is because the set $\{f(x,y_j) | x \in \mathsf{X}\}$ must necessarily have at least two elements (as otherwise there would not have been any uncertainty about $f(x_{\ell,i_0,t_0}, y_j)$), and the map $M_i(y_j, \cdot)$ is one-to-one; so there is at least one value for $x_{\ell,i,t}$ such that $M_i(y_j, f(x_{\ell,i,t}, y_j)) \neq 0$. Thus with constant probability, $\mathit{\Pi}_{(i,t)\neq(i_0,t_0)} R^j_{\ell,i,t} \neq 0$. We take this quantity as z in the second point above, and take π to be $M_{i_0}(y_j, \cdot)$.

Thus, at the end of Phase 1, there exists an index $\widehat{\jmath}$ such that a constant fraction of bundles queried by the adversary in Phase 1 are $\widehat{\jmath}$-undetermined. By Lemma 1, we can conclude that $r_{\widehat{\jmath}}$ is (almost) uniformly distributed.

C Proof of Security of the OTM Protocol

We begin by describing the simulator. To do so formally, we divide the invocations of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ into the following imaginary stages.

– Communication Phase: W.l.o.g., we assume that Bob first schedules the communication session. (Any other strategy is "dominated" by this, in that it can be simulated by an adversary as above.)
– Phase 1: This phase starts when Bob invokes one of the 2κ sessions, and lasts until, for $c_1\kappa$ values of $i \in [\kappa]$, at least one of the sessions $(i, 0)$ and $(i, 1)$ has been invoked. Here, c_1 is a constant to be determined later.
– Phase 2: This phase consists of the remaining invocations of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$.

The simulator handles the above phases as follows:

– Communication Phase: Uniformly pick z_0, z_1, and faithfully simulate all communication sessions using these as the sender's inputs.
– Phase 1: To simulate a new session $(i, \beta) \in [\kappa] \times \{0, 1\}$, accept $\mathbf{j} = j^-$ from Bob, choose $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,m})$ uniformly at random from the appropriate domain, and respond to Bob with $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,j-1}, a^{\beta}_{i,j+1}, a^{\beta}_{i,m})$.
– Input Extraction: At the end of Phase 1, there exists a bit $\bar{b}$ such that there are at least $c_1\kappa/2$ indeces $i \in [\kappa]$, such that the adversary invoked $(i, \bar{b})$ before invoking $(i, 1 - \bar{b})$. The simulator sends $b := 1 - \bar{b}$ to the OT ideal functionality and obtains input s_b.
– Phase 2: The simulator answers the remaining sessions (i, β) by selecting $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,m})$ uniformly at random, conditioned on $r_b = s_b - z_b$.

The proof of security relies on the following properties of the real execution:

1. (r_0, r_1) are uniformly distributed.
2. Conditioned on a particular (r_0, r_1), the sequences $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,j-1}, a^{\beta}_{i,j+1}, \ldots, a^{\beta}_{i,m})$ revealed to the adversary in Phase 1 are uniformly distributed.
3. Let $(b, \bar{b})$ be as defined for the simulator at the end of Phase 1. Then $r_{\bar{b}}$ is almost uniform even given r_b and the adversary's view in both the phases.

The first point follows directly from the fact that R^0_i and R^1_i are independently and uniformly distributed. To see the second point, let $S \subset [\kappa]$ be the set of indeces i, such that neither $(i, 0)$ nor $(i, 1)$ was invoked by the adversary in Phase 1. Note that $|S| = (1 - c_1)\kappa$. We can write $r_\beta = v_\beta + u_\beta$ for $\beta \in \{0, 1\}$, where $u_b = \sum_{i \in S} R^{\beta}_i$. As u_0 and u_1 are independently and uniformly distributed, we have that $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,j-1}, a^{\beta}_{i,j+1}, \ldots, a^{\beta}_{i,m})$ revealed in Phase 1 are independent of r_β.

Now we come to the third point. There exists $\bar{b} \in \{0, 1\}$ such that $c_1\kappa/2$ of the sessions i invoked by the adversary are such that the session $(i, \bar{b})$ was invoked before session $(i, 1 - \bar{b})$. Following notation in previous section, we will say that such an index i is partially-$\bar{b}$-undetermined. Let i be such a partially-$\bar{b}$-undetermined index, and let $j \in [m]$ be the index whose value is *not* revealed to the adversary in session $(i, \bar{b})$. Consider the event $x^{1-\bar{b}}_i = j$, which we will refer to by saying that index i is $\bar{b}$-undetermined. When this event occurs, the adversary remains uncertain about $R_i^{\bar{b}}$. Further, this event occurs with constant probability (namely, $1/m$). Thus, out of the $c_1\kappa/2$ partially-$\bar{b}$-undetermined indeces at the end of Phase 1, $\Omega(\kappa)$ of them are also $\bar{b}$-undetermined. Let $S \subset [\kappa]$ be the set of the $\bar{b}$-undetermined indeces. Then, we can write $r_{\bar{b}} = u_{\bar{b}} + v_{\bar{b}}$, where $u_{\bar{b}} = \sum_{i \in S} R_i^{\bar{b}}$. By Lemma 1, $u_{\bar{b}}$ is almost uniformly distributed, and hence $r_{\bar{b}}$ is almost uniformly distributed.

Non-interactive, Unconditionally Secure Two-Party Computation. Combining the protocol from this section with Theorem 13 from [20], we have the following theorem.

Theorem 2. *Let $f : \mathsf{X} \times \mathsf{Y} \to \mathsf{Z}$ be a constant-size deterministic function, and let $\mathcal{L}$ be a set of functions with domain X, such that $\mathcal{F}^{(f,\mathcal{L})}$ is a non-trivial leaky functionality. Let $g(x, y)$ be a polynomial time, two-party functionality where only the receiver gets the output. Then, there exists a non-interactive, unconditionally secure protocol that securely realized $g(x, y)$ in the $\mathcal{F}^{(f,\mathcal{L})}$-hybrid model.*