(Efficient) Universally Composable Two-Party Computation Using a Minimal Number of Stateless Tokens

Seung Geol Choi* Jonathan Katz* Dominique Schröder*†
Arkady Yerukhimovich* Hong-Sheng Zhou*‡

December 19, 2011

Abstract

We continue the line of work initiated by Katz (Eurocrypt 2007) on using *tamper-proof hardware tokens* for universally composable secure computation with no additional setup. As our main result, we show an efficient oblivious-transfer (OT) protocol in which two parties each create and exchange a single, stateless token and can then run an unbounded number of OTs. This, in turn, means that the parties can perform (repeated) secure computation of arbitrary functions without exchanging additional tokens. Our protocol yields what we believe is the most *practical* and *efficient* known approach for universally composable computation based on tamper-proof hardware.

Motivated by our result, we investigate the minimal number of stateless tokens needed for universally composable secure computation. We prove that our protocol is *optimal* in this regard for constructions having black-box security proofs. We also show that nonblack-box techniques can be used to obtain a construction using only a single stateless token.

*Department of Computer Science, University of Maryland. Email: {sgchoi, jkatz, schroder, arkady, hszhou}@cs.umd.edu. This work is supported in part by DARPA and by NSF awards #0447075 and #1111599.
†Supported in part by a DAAD postdoctoral fellowship.
‡Supported by an NSF CI postdoctoral fellowship.

1 Introduction

The universal composability (UC) framework [Can01] provides a way of analyzing protocols while ensuring strong security guarantees. In particular, protocols proven secure in this framework remain secure when run concurrently with arbitrary other protocols in a larger networked environment. Unfortunately, it soon became clear that most interesting cryptographic tasks are impossible to realize in the "plain" UC framework whenever an honest majority cannot be assumed and, in particular, in the setting of two-party secure computation [CF01, CKL06, Lin09]. This stark negative result has motivated researchers to explore various extensions/variants of the UC framework in which secure computation can be achieved [Can07], with notable examples being the assumption of a common reference string (CRS) [Can01, CF01, CLOS02] or a public-key infrastructure [Can01, BCNP04]. In the real world, implementing either of these approaches seems to require the existence of some *trusted entity* that parties agree to use (though see [CPS07] for some ideas on using a naturally occurring high-entropy source in place of a CRS).

Katz [Kat07] suggested using *tamper-proof hardware tokens* for universally composable computation, with the idea being that the existence of tamper-proof hardware can be viewed as a *physical* assumption rather than an assumption of trust in some external entity. In fact, in Katz's model the parties may create the tamper-proof tokens themselves (rather than obtain them from a trusted provider [HMU05]), and a malicious party can put any algorithm on a token they create. The only assumption is that malicious parties cannot tamper with (or see the internals of) honestly generated tokens. In addition to reducing/removing trust assumptions, secure hardware may also potentially result in more efficient protocols; indeed, it has been suggested for improving efficiency in other settings (e.g., [DQ87, CP93a, CP93b, Bra94, HL08, JKSS10, Kol10, FPS+11, BFSK11]).

In addition to introducing the model, Katz showed that tamper-proof hardware tokens can be used for universally composable computation of arbitrary functions. His work motivated an extensive amount of follow-up work [CGS08, MS08, GKR08, DNW09, GIMS10, GIS+10, DKM11] that we discuss in detail later.

As our main result, we show here a new protocol for universally composable oblivious transfer (OT) based on tamper-proof hardware tokens, implying UC computation of arbitrary functions [CLOS02]. Our work yields what we believe to be the most *practical* and *efficient* known protocol for UC computation based on tamper-proof hardware, since our OT protocol simultaneously achieves all the following properties (which are not achieved all at once by any other solution; see Table 1 for a detailed comparison):

- Our protocol is based on *stateless* tokens, which seem easier/cheaper to create in practice and are resistant to resetting attacks.

- Our protocol requires the parties to exchange a *single* pair of tokens. This can be done in advance of running the protocol, before the parties' inputs are known. Furthermore, the tokens can be used to implement an *unbounded* number of oblivious transfers, rather than requiring the parties to exchange a fresh pair of tokens for every oblivious transfer they want to compute. Thus, the parties can use the same tokens to perform an unlimited number of secure computations (of possibly different functions, and on different inputs).

- Our protocol is quite efficient: It is fully black-box, and can be based on standard symmetric-key primitives (namely, block ciphers and collision-resistant hash functions) rather than

public-key techniques. Moreover, any desired number of OTs can be obtained (in parallel) in *constant* rounds.[1]

Inspired by our result, we turn next to investigating the minimal number of stateless tokens needed for UC secure computation. We show that two tokens — one created by each party — are needed even to obtain a single universally composable OT as long as only "black-box techniques" are used. (We explain what we mean by "black-box techniques" in the relevant section of our paper.) Our protocol, above, is thus optimal in this regard. Our impossibility result is somewhat surprising, since a single *stateful* token suffices for OT [DKM11]. Our results thus demonstrate an inherent difference between stateful and stateless tokens.

Since protocols based on nonblack-box techniques tend to be impractical, our work pins down the minimal number of stateless tokens needed as far as practical protocols are concerned. From a theoretical point of view, however, it is still interesting to completely resolve the question. In this vein, we show a protocol for carrying out an unbounded number of secure computations using only a *single* (stateless) token. Our construction uses a variant of the nonblack-box simulation technique introduced by Barak [Bar01].

In summary, our work shows that highly efficient, universally composable computation can be realized from two stateless tokens, and is unlikely using a single stateless token. On the other hand, UC computation from a single stateless token can be achieved using nonblack-box techniques.

1.1 Related Work

Katz's original protocol for secure computation using tamper-proof tokens [Kat07] required stateful tokens and relied on number-theoretic assumptions (specifically, the DDH assumption). Subsequent work has mainly focused on improving one or both of these aspects of his work.

Several researchers have explored constructions using *stateless* tokens. Stateless tokens are presumably easier (and/or cheaper) to build, and are resistant to resetting attacks whereby an adversary cuts off the power supply and thus effectively "rewinds" the token. Chandran et al. [CGS08] were the first to eliminate the requirement of stateful tokens. They construct commitments based on the existence of one-way functions, and oblivious transfer based on any enhanced trapdoor permutation (eTDP). From a practical perspective, however, their work has several drawbacks. Their OT protocol makes nonblack-box use of the underlying primitives, runs in $\Theta(\lambda)$ rounds (where λ is the security parameter), and uses the heavy machinery of concurrent non-malleable zero-knowledge. Improving upon their work, Goyal et al. [GIS$^+$10] show a black-box construction of oblivious transfer; their protocol runs in constant rounds assuming a collision-resistant hash function, or $\Theta(\lambda/\log\lambda)$ rounds based on one-way functions. However, their protocol requires the parties to exchange $\Theta(\lambda)$ tokens for *every* oblivious transfer the parties wish to execute.

A second direction has explored the possibility of eliminating computational assumptions altogether. This line of work was initiated by Moran and Segev [MS08], who showed how to realize statistically secure UC commitments using a single stateful token. (Note that commitment does not imply OT, or general secure computation, in the unconditional setting.) Their construction can be used for any bounded number of commitments, still using only one token, and the authors note that they can achieve an unbounded number of commitments (with computational security) based on one-way functions. Goyal et al. [GIS$^+$10] show an unconditional construction of oblivious

[1] If desired, our protocol can also be based on one-way functions (this assumption is minimal for OT based on stateless tokens [GIMS10]), at the expense of increasing the round complexity.

	stateless tokens			stateful tokens		
	Here	[CGS08]	[GIS$^+$10]	[Kat07]	[GIS$^+$10]	[DKM11]
Tokens:	2	2	$\Theta(\lambda)$	2	$\Theta(\lambda)$	1
Rounds:	$\Theta(1)$	$\Theta(\lambda)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Assumptions:	CRHF	eTDP	CRHF	DDH	none	none
# of OTs:	unbounded	unbounded	1	unbounded	1	bounded

Table 1: Universally composable OT based on tamper-proof hardware tokens. The security parameter is denoted by λ. A variant of our protocol (and that of [GIS$^+$10]) can be based on one-way functions, but using $\Theta(\lambda/\log\lambda)$ rounds.

transfer (and hence general secure computation) using $\Theta(\lambda)$ stateful tokens. Recently, Döttling et al. [DKM11] show how to construct unconditionally secure OT using only a single stateful token. Goyal et al. [GIMS10] showed that unconditional security from stateless tokens is impossible (unless the token model is extended to allow tokens to encapsulate each other).

Kolesnikov [Kol10] showed an efficient construction of oblivious transfer from stateless tokens. However, his work is not in the UC setting, and he achieves covert security [AL07] rather than security against malicious parties. Dubovitskaya et al. [DSV10] constructed an OT protocol from two stateful tokens. Their work assumes tokens are not reused (it is not clear how this is enforced), and is also not in the UC setting.

Our work in relation to prior work. For our main result, we carefully combine the techniques of [GIS$^+$10] and [DKM11] to achieve the most *practical* and *efficient* known protocol for universally composable OT (and hence general secure computation) based on tamper-proof hardware tokens. Our protocol uses two *stateless* tokens (one per party), can be used for an unbounded number of OTs, runs in constant rounds, and could be instantiated quite efficiently based on standard symmetric-key primitives such as SHA-1 and AES. A detailed comparison of this protocol to relevant prior work is given in Table 1.

In addition to the above, we show two other results: there is no "black-box" construction of universally composable OT using fewer than two stateless tokens, but universally composable con tossing (and hence OT *can* be based on a single stateless token using nonblack-box techniques.

In independent work, Döttling et al. [DMMQN11] show a different nonblack-box construction of a UC coin-tossing protocol based on a single stateless hardware token, and briefly justify why nonblack-box techniques are needed. In our work we provide a rigorous proof of this intuition. Our efficient, black-box OT protocol using two stateless tokens has no analog in their work.

2 Background and Cryptographic Primitives

In this section, we review the underlying primitives and necessary definitions. In the following definitions all adversaries are assumed to be non-uniform.Let λ be the security parameter. For a set S, we let $x \leftarrow S$ denote choosing x uniformly at random from S.

2.1 Pseudorandom Functions

A function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{\ell(\lambda)} \mapsto \{0,1\}^{\ell(\lambda)}$ with a polynomial ℓ is pseudorandom if for every polynomial time adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\left|\Pr[\mathcal{A}^{\mathsf{PRF}_k(\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{H(\cdot)}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where $k \leftarrow \{0,1\}^\lambda$ is chosen uniformly at random and H is chosen uniformly at random from the set of functions mapping $\ell(\lambda)$-bit strings to $\ell(\lambda)$-bit strings. Pseudorandom functions can be constructed in a black-box way from any one-way function [HILL99, GGM86]. In our constructions, the polynomial ℓ is chosen to be sufficiently large so that we can always truncate the output to achieve a pseudorandom string of the necessary length.

2.2 Message Authentication Codes

A message authentication code $\mathsf{MAC} = (\mathsf{Kg}, \mathsf{Tag}, \mathsf{Vrfy})$ is a triple of efficient algorithms where

Key Generation. $s \leftarrow \mathsf{Kg}(1^\lambda)$ returns a key s.

Authentication. The authentication algorithm $\tau \leftarrow \mathsf{Tag}_s(m)$ takes as input the key s, a message m from a message space $\mathcal{M}_\lambda$ and returns a tag τ in a range $\mathcal{R}_\lambda$.

Verification. $\mathsf{Vrfy}_s(m, \tau)$ returns a bit.

It is assumed that the scheme is complete, i.e., for all $s \leftarrow \mathsf{Kg}(1^\lambda)$, any $m \in \mathcal{M}_\lambda$, and any $\tau \leftarrow \mathsf{Tag}_s(m)$ we have $\mathsf{Vrfy}_s(m, \tau) = 1$. A message authentication code is deterministic if the tagging algorithm is deterministic. A deterministic MAC is canonical if the verification algorithm recomputes the tag for a given message and checks that they match, i.e., $\mathsf{Vrfy}_s(m, \tau)$ outputs 1 if $\mathsf{Tag}_s(m) = \tau$; otherwise it outputs 0.

Unforgeability of a message authentication codes demands that it is infeasible to produce a valid tag for a new message even for an adversary allowed to ask for tags on messages of its choice.

Definition 1 A message authentication code $\mathsf{MAC} = (\mathsf{Kg}, \mathsf{Tag}, \mathsf{Vrfy})$ is unforgeable under chosen message attacks (EU-CMA) if for any polynomial time algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{MAC}}$ evaluates to 1 is negligible (as a function of λ), where

Experiment $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{MAC}}(\lambda)$
$s \leftarrow \mathsf{Kg}(1^\lambda)$
$(m^*, \tau^*) \leftarrow \mathcal{A}^{\mathsf{Tag}_s(\cdot), \mathsf{Vrfy}_s(\cdot,\cdot)}(1^\lambda)$
Return 1 if $\mathcal{A}$ returns a pair m^*, τ^* such that
$\mathsf{Vrfy}_s(m^*, \tau^*) = 1$ and $\mathcal{A}$ has never queried $\mathsf{Tag}_s(m^*)$ before.

The probability is taken over all coin tosses of $\mathsf{Kg}, \mathsf{Tag}$, and Vrfy.

Message authentication codes for arbitrary length messages can be constructed in a black-box way from any one-way function [GGM85].

2.3 Commitment Schemes

A commitment scheme consists of a pair of probabilistic, interactive algorithms (S, R).

1. In the *commit* phase, the inputs to S are a message m, and a random tape r_m, while the input to R is a random tape $\breve{r}_m$. We denote the protocol execution by $\mathsf{Com}(m; r_m, \breve{r}_m) := \mathsf{trans}\langle \mathsf{S}(m, r_m), \mathsf{R}(\breve{r}_m)\rangle$. When it is clear from the context, we sometimes omit the randomness of the receiver and just write $\mathsf{Com}(m; r_m)$. Let com_m be the resulting transcript (i.e., $\mathsf{com}_m = \mathsf{Com}(m; r_m, \breve{r}_m)$).

2. Without loss of generality, we assume a *canonical decommit* phase in which the sender sends the message m together with the randomness r_m that was used to generate the commitment (i.e., $\mathsf{decom}_m = (m, r_m)$). Then, the receiver runs the algorithm $\mathsf{Open}(\mathsf{com}_m, m, r_m)$ which checks if (m, r_m) is consistent with the transcript com_m. In particular, let $\mathsf{com}_m = (\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_\ell)$ for some ℓ where α_i and β_i are the messages from S and R respectively (if R sends the message first, α_1 is set to $\perp$). Then, the algorithm Open checks if $\alpha_i = \mathsf{S}(m, r_m, (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}))$ for $i = 0, \ldots, \ell - 1$. If so, Open outputs m; otherwise it outputs $\perp$.

Secure commitment schemes must satisfy two properties: hiding and binding. We first review the hiding property of commitment schemes. Hiding means that no adversary can distinguish which of two messages are locked in the commitment.

Hiding. Consider a commitment scheme $\mathcal{C} = (\mathsf{S}, \mathsf{R})$. Given an adversary R^*, let $\mathsf{view}_{\mathsf{R}^*}(m)$ denote the distribution on the view of R^* when executing the commit phase protocol with the sender S committing to a message $m \in \{0,1\}^\lambda$. This distribution is taken over the random coins of S and R^*, that is

$$\mathsf{view}_{\mathsf{R}^*}(m) := \{r_m, \breve{r}_m \leftarrow \{0,1\}^* : \ \mathsf{view}_{\mathsf{R}^*}\langle \mathsf{S}(m; r_m), \mathsf{R}^*(\breve{r}_m)\rangle\}.$$

Definition 2 $\mathcal{C}$ is *computationally hiding* if for any messages $m_0, m_1 \in \{0,1\}^\lambda$ and for any polynomial time algorithm R^*, it holds that

$$|\Pr[v \leftarrow \mathsf{view}_{\mathsf{R}^*}(m_0); \mathsf{R}^*(v) = 1] - \Pr[v \leftarrow \mathsf{view}_{\mathsf{R}^*}(m_1); \mathsf{R}^*(v) = 1]| \leq \mathsf{negl}(\lambda).$$

Definition 3 $\mathcal{C}$ is *statistically hiding* if for any message $m_0, m_1 \in \{0,1\}^\lambda$ and any algorithm R^*, the statistical distance between the distributions $\mathsf{view}_{\mathsf{R}^*}(m_0)$ and $\mathsf{view}_{\mathsf{R}^*}(m_1)$ is negligible in λ.

Binding. Binding means that the adversary cannot open the commitment in two different ways.

Definition 4 Let $\mathcal{C} = (\mathsf{S}, \mathsf{R})$ be a commitment scheme and com be the transcript from the execution of a commit phase protocol between a (malicious) sender S^* and the receiver R, that is $\mathsf{com} = \mathsf{trans}\langle \mathsf{S}^*(r_m), \mathsf{R}(\breve{r}_m)\rangle$ with $r_m, \breve{r}_m \leftarrow \{0,1\}^*$. Commitment scheme $\mathcal{C}$ is *computationally* (resp., *statistically*) *binding* if for any polynomial time (resp., arbitrary) adversary S^*, the probability that S^* outputs a tuple $(m_0, m_1, r_{m_0}, r_{m_1})$ such that $m_0 = \mathsf{Open}(\mathsf{com}, m_0, r_{m_0})$ and $m_1 = \mathsf{Open}(\mathsf{com}, m_1, r_{m_1})$ with $m_0 \neq m_1$ is negligible in λ.

Here, we also need a slightly stronger definition. Following [HK05] we say that a commitment scheme is strongly binding, if the probability (over the coin tosses $\breve{r}$ of the honest receiver) that there exists distinct messages m_0, m_1 and coins r_{m_0}, r_{m_1} for S^* such that the corresponding commitments to m_0, m_1 are identical is at most negligible in λ.

Definition 5 $\mathcal{C}$ is *strongly binding* if for all messages $m_0, m_1 \in \{0,1\}^\lambda$ it holds that

$$\Pr_{\breve{r}}\left[\begin{array}{l} \exists \text{ distinct } m_0, m_1 \in \{0,1\}^\lambda, r_{m_0}, r_{m_1} \text{ such that} \\ \mathsf{Com}(m_0; r_{m_0}, \breve{r}) = \mathsf{Com}(m_1; r_{m_1}, \breve{r}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We use Scom to denote a (possibly interactive) statistically-hiding and computationally-binding commitment scheme and Com to denote a (possibly interactive) computationally-hiding and strongly-binding commitment scheme. We let $\mathsf{com}_m \leftarrow \mathsf{Com}(m; r_m, \breve{r}_m)$ denote a commitment to a message m, where the sender (resp., receiver) uses uniform random coins r_m (resp., $\breve{r}_m$) and the final transcript is com_m;[2] sometimes we omit $\breve{r}_m$ when it is clear from the context. (We also use similar notation for Scom.)

Instantiations. In our constructions, we use two types of commitment schemes: a computationally-hiding and strongly-binding commitment and a statistically-hiding and computationally-biding commitment. The former can be instantiated with a non-interactive protocol from any one-to-one one-way function [Blu82] or with a constant-round protocol from any one-way function by Naor [Nao91]. The latter can be instantiated with a constant-round protocol from collision-resistant hash functions [DPP97], or with a $O(\lambda/\log\lambda)$-round protocol from any one-way function [HR07].

2.4 Strong Randomness Extractors

An efficiently computable function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ is a strong (k,ϵ)-extractor if for every source X with min-entropy at least k, the statistical distance between $(U_d, \mathsf{Ext}(X, U_d))$ and (U_d, U_ℓ) is at most ϵ, where U_d and U_ℓ are uniform distributions of length d and ℓ respectively.

Theorem 1 (Leftover Hash Lemma [HILL99]) *If the family $\mathcal{H}$ of hash functions $h : \{0,1\}^n \to \{0,1\}^\ell$ is pairwise independent with $\ell = k - 2\log(1/\epsilon) - O(1)$, then $\mathsf{Ext}(X,h) := h(X)$ is a strong $(k, \epsilon/2)$-extractor.*

In our constructions, the source X will have min-entropy at least $2\lambda - O(\log\lambda)$, and we set ℓ equal to λ. Thus, we can make ϵ negligible in λ (e.g., at most $2^{-\lambda/3}$).

2.5 A Family of Collision Resistant Hash Functions

$\mathcal{H} = \{h_k : \{0,1\}^* \to \{0,1\}^{\ell(\lambda)}\}_{k\in\{0,1\}^\lambda}$ with a polynomial ℓ is a family of collision resistant hash functions if for any non-uniform PPT algorithm $\mathcal{A}$, it holds that

$$\Pr[k \leftarrow \{0,1\}^\lambda : \mathcal{A}(k) = (x_1, x_2) \text{ s.t. } x_1 \neq x_2 \text{ and } h_k(x_1) = h_k(x_2)] \leq \mathsf{negl}(\lambda).$$

2.6 Witness-Indistinguishable Arguments of Knowledge

For interactive algorithms A and B, let $\langle A(y), B(z)\rangle(x)$ denote a random variable representing the execution between A and B on a public input x when A's private input is y and B's private input is z. Let $\mathsf{out}_A\langle A(y), B(z)\rangle(x)$ and $\mathsf{view}_A\langle A(y), B(z)\rangle(x)$ denotes the output and the view of A in this execution. We define out_B and view_B symmetrically. Let $\mathsf{trans}\langle A(y), B(z)\rangle(x)$ denote transcript of the execution.

Definition 6 A pair of interactive PPT algorithms (P, V) is called witness-indistinguishable (WI) argument system for a language $L \in \mathcal{NP}$ if it satisfies the following properties:

[2]Since this notation implicitly defines how r_m and $\breve{r}_m$ are sampled, hereafter we will use the notation without explicitly mentioning how to sample these values.

Completeness: For any λ, any $x \in L$, and any witness w for x, it holds that

$$\Pr[\mathsf{out}_V\langle P(1^\lambda, w), V(1^\lambda)\rangle(x) = 1] = 1.$$

Witness indistinguishability: For any non-uniform PPT algorithm V^* and any sequence $\{(x_\lambda, w_\lambda, w'_\lambda)\}_{\lambda\in\mathbb{N}}$ where $|x_\lambda| = \lambda$ and w_λ, w'_λ are witnesses for $x_\lambda \in L$, the following two probability ensembles are computationally indistinguishable:

$$\left\{\mathsf{view}_{V^*}\langle P(1^\lambda, w_\lambda), V^*(1^\lambda)\rangle(x_\lambda)\right\}_{\lambda\in\mathbb{N}} \approx \left\{\mathsf{view}_{V^*}\langle P(1^\lambda, w'_\lambda), V^*(1^\lambda)\rangle(x_\lambda)\right\}_{\lambda\in\mathbb{N}}$$

Argument of knowledge: There exists a non-uniform PPT algorithm E (called *knowledge extractor*) such that for any non-uniform PPT algorithm P^* and for any x, it holds that

$$\Pr[E(P^*, x) \text{ outputs a witness certifying } x \in L] \geq \Pr[\mathsf{out}_V\langle P^*(1^\lambda), V(1^\lambda)\rangle(x) = 1] - \mathsf{negl}(\lambda).$$

Instantiation. A constant round WI argument of knowledge for any $L \in \mathcal{NP}$ can be constructed from one-way functions [FS90].

2.7 Linear Algebra

If $a \in \mathbb{F}_2^\lambda$ and $b \in \mathbb{F}_2^k$ are two column-vectors, then $(ab^{\mathrm{T}}) = (a_i b_j)_{ij} \in \mathbb{F}_2^{\lambda\times k}$ is the outer product (or tensor product) of a and b, and $a^{\mathrm{T}}b = \sum_{i=1}^{\lambda} a_i b_i \in \mathbb{F}_2$ the inner product.

Let $C \in \mathbb{F}_2^{\lambda\times 2\lambda}$. Then $\dim(\ker(C)) \geq \lambda$. Let $B = \{b_1, \ldots, b_\lambda\} \subseteq \ker(C)$ be a linearly independent set. One can choose a set $B^* = \{b_{\lambda+1}, \ldots, b_{2\lambda}\}$ such that $B \cup B^*$ is a basis of $\mathbb{F}_2^{2\lambda}$. Let $e_i \in \mathbb{F}_2^\lambda$ be the ith unit-vector. Then, there exists a matrix $G \in \mathbb{F}_2^{\lambda\times 2\lambda}$ such that $Gb_i = e_i$ for $i = 1, \ldots, \lambda$ and $Gb_i = 0$ for $i = \lambda+1, \ldots, 2\lambda$. This matrix is called the *complementary matrix* of C and we denote by $G \leftarrow \mathsf{Comp}(C)$ its computation. It holds that $\mathrm{rank}(G) = \lambda$ and $B^* \subseteq \ker(G)$. For such C and G, we can always solve the linear system $Cx = r$, $Gx = s$ by solving $Cx_{B^*} = r$ and $Gx_B = s$ independently with $x_B \in \mathrm{span}(B) \subseteq \ker(C)$ and $x_{B^*} \in \mathrm{span}(B^*) \subseteq \ker(G)$, and then setting $x := x_B + x_{B^*}$.

3 Ideal Functionalities

We present the ideal functionalities for stateful, tamper-proof hardware tokens in Figure 1, for (single-instance) oblivious transfer in Figure 2, and for (single-instance) coin tossing in Figure 3.

Token functionality. We model tamper-proof hardware tokens as an ideal functionality in the UC framework, following Katz [Kat07]; see Figure 1. Our ideal functionality models stateful tokens: although all our protocols use stateless tokens, an adversarially generated token may be stateful.

Oblivious-transfer functionality. The OT functionality is standard, but we wish here to model a multi-session variant where the sender and receiver repeatedly (in different sub-sessions) execute any agreed-upon number m of parallel OTs (in a given sub-session). We refer to this functionality as $\mathcal{F}_{\mathsf{multi\text{-}OT}}$, and describe it in Figure 4. As highlighted in [GIS+10], the sender is notified each time the receiver obtains output.

Coin-tossing functionality. We also use the coin-tossing functionality $\mathcal{F}_{\mathsf{coin}}$, defined in the natural way; see Figure 3.

Functionality $\mathcal{F}_{\mathsf{wrap}}$

The functionality is parameterized by a polynomial $p(\cdot)$ and an implicit security parameter λ.

Create:
Upon receiving an input $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle, \mathcal{M})$ from a party C (i.e., the token creator), where U is another party (i.e., the token user) in the system and $\mathcal{M}$ is an interactive Turing machine, do:

> If there is no tuple of the form $\langle \mathsf{C}, \mathsf{U}, \star, \star, \star \rangle$ stored, store $\langle \mathsf{C}, \mathsf{U}, \mathcal{M}, 0, \emptyset \rangle$.
> Reveal $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ to the adversary.

Upon receiving a message $(\text{READY}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ from the adversary, send $(\text{READY}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ to U.

Execute:
Upon receiving an input $(\text{RUN}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle, \mathsf{msg})$ from the user U, find the unique stored tuple $\langle \mathsf{C}, \mathsf{U}, \mathcal{M}, i, \mathsf{state} \rangle$. If no such tuple exists, do nothing. Otherwise, do:

> If the Turing machine $\mathcal{M}$ has never been used yet, i.e., $i = 0$, then choose ω uniformly at random from $\{0,1\}^{p(\lambda)}$ and set $\mathsf{state} := \omega$ before running the Turing machine.
> Run $(\mathsf{out}, \mathsf{state}') \leftarrow \mathcal{M}(\mathsf{msg}; \mathsf{state})$ for at most $p(\lambda)$ steps where out is the response and state' is the new state of $\mathcal{M}$ (set $\mathsf{out} := \perp$ and $\mathsf{state}' := \mathsf{state}$ if $\mathcal{M}$ does not respond in the allotted time). Send $(\text{RESPONSE}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle, \mathsf{out})$ to U. Erase $\langle \mathsf{C}, \mathsf{U}, \mathcal{M}, i, \mathsf{state} \rangle$ and store $\langle \mathsf{C}, \mathsf{U}, \mathcal{M}, i+1, \mathsf{state}' \rangle$.

Figure 1: The ideal $\mathcal{F}_{\mathsf{wrap}}$ functionality for stateful tokens.

Functionality $\mathcal{F}_{\mathsf{OT}}$

$\mathcal{F}_{\mathsf{OT}}$ interacts with sender S, receiver R, and the adversary.

Upon receiving an input $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \langle x_0, x_1 \rangle)$ from S with $x_0, x_1 \in \{0,1\}$, record the tuple $\langle x_0, x_1 \rangle$, and reveal $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to the adversary. Ignore further $(\text{SEND}, \ldots)$ inputs.

Upon receiving an input $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, b)$ from R with $b \in \{0,1\}$, record the bit b, and reveal $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to the adversary. Ignore further $(\text{RECEIVE}, \ldots)$ inputs.

Upon receiving a message $(\text{GO}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ from the adversary, ignore the message if $\langle x_0, x_1 \rangle$ or b is not recorded. Otherwise, do the following: return $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to S, and return $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, x_b)$ to R. Ignore further $(\text{GO}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ messages from the adversary.

Figure 2: The ideal $\mathcal{F}_{\mathsf{OT}}$ functionality for bit-OT.

4 Efficient Oblivious Transfer Using Two Stateless Tokens

We provide some intuition and background before giving the details of our protocol. Our starting point is the unconditionally secure OT protocol from [DKM11], which uses a single *stateful* token. We sketch a simplified version of their protocol for the case of a single OT carried out between the sender S with input $(x_0, x_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ and the receiver R with input $b \in \{0,1\}$. By $\mathbb{F}_2$ we denote the finite field with two elements, and we canonically identify the vector space $\mathbb{F}_2^\lambda$ with the set $\{0,1\}^\lambda$ of strings of length λ. The main steps of the protocol are as follows:

1. S generates a token $\mathcal{T}_{\mathsf{S}}$ holding random vector $a \in \mathbb{F}_2^{2\lambda}$ and matrix $B \in \mathbb{F}_2^{2\lambda \times 2\lambda}$ and gives it to R.

2. R chooses a random matrix $C \in \mathbb{F}_2^{\lambda \times 2\lambda}$ and sends it to S. In turn, S computes $\tilde{a} := Ca$,

Functionality $\mathcal{F}_{\mathsf{coin}}$

$\mathcal{F}_{\mathsf{coin}}$ interacts with two parties, Alice A and Bob B, and the adversary. The functionality is parameterized by a security parameter λ. It also maintains variables $(b_{\mathsf{A}}, b_{\mathsf{B}}, \mathsf{coins})$ initialized to $(\mathsf{false}, \mathsf{false}, \perp)$.

Upon receiving an input $(\text{TOSS}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle)$ from party $P \in \{\mathsf{A}, \mathsf{B}\}$, then set $b_P := \mathsf{true}$, and reveal $(\text{TOSS}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, P)$ to the adversary. Ignore further $(\text{TOSS}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle)$ inputs from the party P.

Upon receiving a message $(\text{GO}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, P)$ from the adversary for $P \in \{\mathsf{A}, \mathsf{B}\}$, ignore the message if $b_{\mathsf{A}} = \mathsf{false}$ or $b_{\mathsf{B}} = \mathsf{false}$. Otherwise, do the following: if $\mathsf{coins} = \perp$, i.e, coins has not been set yet, then randomly choose $u \leftarrow \{0,1\}^\lambda$ and set $\mathsf{coins} := u$; return $(\text{COIN}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins})$ to party P. Ignore further $(\text{GO}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, P)$ messages for the party P from the adversary.

Figure 3: The ideal $\mathcal{F}_{\mathsf{coin}}$ functionality for coin tossing.

Functionality $\mathcal{F}_{\mathsf{multi\text{-}OT}}$

$\mathcal{F}_{\mathsf{multi\text{-}OT}}$ interacts with sender S, receiver R, and the adversary. The functionality is parameterized by a security parameter λ. It also maintains a variable $\mathsf{curr\text{-}id}$ initialized to $\perp$.

Upon receiving an input $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \langle m, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle)$ from S with all $x_i^0, x_i^1 \in \{0,1\}^\lambda$, if $\mathsf{curr\text{-}id} \notin \{\perp, \mathsf{ssid}\}$ then ignore the input. Otherwise, set $\mathsf{curr\text{-}id} := \mathsf{ssid}$ and record $\langle \mathsf{ssid}, m, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle$, and reveal $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to the adversary. Ignore further $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \ldots)$ inputs with this ssid.

Upon receiving an input $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \langle m, \{b_i\}_{i=1}^m \rangle)$ from R with all $b_i \in \{0,1\}$, if $\mathsf{curr\text{-}id} \notin \{\perp, \mathsf{ssid}\}$ then ignore the input. Otherwise, set $\mathsf{curr\text{-}id} := \mathsf{ssid}$ and record the tuple $\langle \mathsf{ssid}, m, \{b_i\}_{i=1}^m \rangle$, and reveal $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to the adversary. Ignore further $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \ldots)$ inputs with this ssid.

Upon receiving a message $(\text{GO}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ from the adversary, ignore the message if $\mathsf{curr\text{-}id} \neq \mathsf{ssid}$, or either $\langle \mathsf{ssid}, m, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle$ or $\langle \mathsf{ssid}, m, \{b_i\}_{i=1}^m \rangle$ is not recorded. Otherwise, do the following: set $\mathsf{curr\text{-}id} := \perp$, return $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to S, and return $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{x_i^{b_i}\}_{i=1}^m)$ to R. Ignore further $(\text{GO}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ messages with this ssid from the adversary.

Figure 4: The $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ functionality.

$\tilde{B} := CB$, and a complementary matrix $G \in \mathbb{F}_2^{\lambda \times 2\lambda}$ to C and sends these to R.

3. R sends a random $h \in \mathbb{F}_2^{2\lambda}$ to S. In turn, S sends $\tilde{x}_0 := x_0 + GBh$ and $\tilde{x}_1 := x_1 + GBh + Ga$ to R.

4. R queries the token with a random $z \in \mathbb{F}_2^{2\lambda}$ such that $z^{\mathrm{T}} h = b$. The token will in turn output $V := az^{\mathrm{T}} + B$. Then R checks that $CV = \tilde{a}z^{\mathrm{T}} + \tilde{B}$ and, if this is the case, outputs $x_b := \tilde{x}_b - GVh$. (Otherwise it detects that S was cheating.)

The basic idea of their protocol is as follows. The receiver R performs a secret sharing of its input b into shares z and h; by using only h with the sender S and only z with the token, R maintains the privacy of b. In order to obtain the output x_b, the receiver R has to compute the mask (i.e., GBh or $GBh + Ga$). This is achieved by querying the token with z, since $GVh = G(az^{\mathrm{T}} + B)h = b(Ga) + GBh$. To ensure that the token outputs the correct value V, the receiver checks that $CV = \tilde{a}z^{\mathrm{T}} + \tilde{B}$. Since the token does not know C, incorrect behavior is detected with overwhelming probability. To achieve security against a malicious receiver, it is crucial that the

receiver is only able to query the token once, and this is enforced by making the token *stateful*. In particular, the token "self destructs" after the first query by the receiver.

Using stateless tokens. We carefully combine the techniques of [DKM11] with ideas from [GIS+10] so that we can use two *stateless* tokens instead of one stateful token. This entails several difficulties:

Multiple queries. To use the ideas of [GIS+10], we need to ensure that a malicious party only queries the token once. Motivated by similar techniques in [GIS+10], we handle this issue by modifying the token so that it only replies to *authenticated* inputs. That is, instead of querying z directly to the token, R queries $(\mathsf{com}_z, z, r_z, \tau_z)$ where com_z is a commitment to z, the value τ_z is a tag on com_z (received from S), and z, r_z is the decommitment of com_z (see the discussion below about why this technique is useful).

Extracting the inputs. Using a stateless token, however, introduces the difficulty of extracting the sender's inputs during the simulation. Extraction from a stateful token is possible by having the simulator rewind the token and query it multiple times. (The fact that the simulator can query the token multiple times is an advantage of the simulator over the real-world parties.) Once we move to a stateless token in the real world, and authenticate the queries as described above, even the simulator is no longer able to rewind and send multiple (authenticated) queries to the token.

To resolve this issue we introduce a second stateless token $\mathcal{T}_{\mathsf{R}}$ sent from the receiver to the sender which allows directly extracting the sender's inputs.

At a high level, the above result in the following changes to the protocol:

1. The receiver sends a token $\mathcal{T}_{\mathsf{R}}$ to the sender. (The behavior of this token will become clear below.)

2(a). S generates a statistically hiding[3] commitment $\mathsf{com} \leftarrow \mathsf{Scom}(a\|B)$ and sends com to R. Then, R chooses a matrix C and authenticates com by computing $\tau := \mathsf{Tag}_s(\mathsf{com}\|C)$ (where s is a secret key also embedded in $\mathcal{T}_{\mathsf{R}}$), and sends C and τ to S.

2(b). S queries $\mathcal{T}_{\mathsf{R}}$ on input $(C, \mathsf{com}, \mathsf{decom}, \tau)$. The token checks that the tag τ is valid and that decom is a valid opening of com; otherwise it aborts. $\mathcal{T}_{\mathsf{R}}$ returns $\tilde{a} := Ca$ and $\tilde{B} := CB$ together with a MAC tag $\tau' := \mathsf{Tag}_s(\tilde{a}\|\tilde{B})$. Then S sends $(\tilde{a}, \tilde{B}, \tau')$ to R.

Intuitively, the value (a, B) remains hidden from a corrupted R as before. For a corrupted sender, the simulator will emulate $\mathcal{F}_{\mathsf{wrap}}$ and observe the inputs to $\mathcal{T}_{\mathsf{R}}$. To guarantee that the simulator extracts the sender's inputs correctly, it is necessary that the sender queries the token exactly once with a value (a, B). The binding property of the commitment scheme, together with the unforgeability of the MAC, guarantee that S can make at most one valid query to the token. The unforgeability of MAC assures that S must query the token at least once to generate a valid next message of the protocol. A similar argument holds for the inputs $(\mathsf{com}_z, z, r_z, \tau_z)$ to the sender's token $\mathcal{T}_{\mathsf{S}}$.

There is one additional issue to be taken care of. The above protocol is secure assuming the tokens are generated as specified by the protocol. However, a malicious token may try to leak some

[3] We were not able to prove security using a computationally hiding commitment scheme. Indeed, it is an interesting open problem to achieve a constant-round OT protocol with two stateless tokens based only on one-way functions.

On input $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \tau_z)$ do:
  $v := \mathsf{Vrfy}_{s_2}(\mathsf{ssid}\|i\|\mathsf{com}_z, \tau_z)$
  if $z = \mathsf{Open}(\mathsf{com}_z, z, r_z)$ and $v = 1$
    $a := \mathsf{PRF}_{k_a}(\mathsf{ssid}\|i)$; $B := \mathsf{PRF}_{k_B}(\mathsf{ssid}\|i)$
    $w := \mathsf{PRF}_{k_w}(\mathsf{ssid}\|i)$; $r_w := \mathsf{PRF}_{k_W}(\mathsf{ssid}\|i)$
    $V := az^{\mathrm{T}} + B$
    output (V, w, r_w)
  else output $(\perp, \perp, \perp)$

Figure 5: The Turing machine $\mathcal{M}_{\mathsf{S}}$ to be embedded in the sender-created token $\mathcal{T}_{\mathsf{S}}$.

On input $(\mathsf{ssid}, i, C, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ do:
  $s := \mathsf{PRF}_{k_s}(\mathsf{ssid})$
  $v := \mathsf{Vrfy}_s(\mathsf{ssid}\|i\|C\|\mathsf{com}_{a\|B}, \tau_{a\|B})$
  if $a\|B = \mathsf{Open}(\mathsf{com}_{a\|B}, a\|B, r_{a\|B})$ and $v = 1$
    $\tilde{a} := Ca$; $\tilde{B} := CB$; $\tau_{\tilde{a}\|\tilde{B}} := \mathsf{Tag}_s(\mathsf{ssid}\|i\|\tilde{a}\|\tilde{B})$
    output $(\tilde{a}, \tilde{B}, \tau_{\tilde{a}\|\tilde{B}})$
  else output $(\perp, \perp, \perp)$

Figure 6: The Turing machine $\mathcal{M}_{\mathsf{R}}$ to be embedded in the receiver-created token $\mathcal{T}_{\mathsf{R}}$.

information about the other party's queries to the token creator. For example, a malicious token $\mathcal{T}_{\mathsf{R}^*}$ may output $(\tilde{a}, \tilde{B}, \tau')$ where the bits of τ' leak information about (a, B). To protect against this, we have the sender first send a commitment U to the output of the token; the receiver then sends some sort of proof regarding the correct behavior of the token (in this case, the actual key for MAC). After the sender verifies that the output is correct, it opens the commitment U. A similar idea is applied to the other token $\mathcal{T}_{\mathsf{S}}$. This technique ensures that the only information leakage that can occur is for the token to abort. For $\mathcal{T}_{\mathsf{S}}$, this type of leakage is already handled by the protocol of [DKM11]. For $\mathcal{T}_{\mathsf{R}}$, we handle this type of leakage using strong extractors; see the formal description of the protocol, below.

The above suffices for a single OT from each pair of tokens. To achieve an unbounded number of OTs we replace all the secret keys and secret inputs with pseudorandom values output by a pseudorandom function.

4.1 The Protocol

The protocol π between a sender S and a receiver R consists of an initial token-exchange phase after which the parties can carry out an unlimited number of oblivious transfers. We describe π now; see also Figure 7.

Token-exchange phase. Each party generates a single token and sends it to the other party. The sender's token $\mathcal{T}_{\mathsf{S}}$ encapsulates the code described in Figure 5, where k_a, k_B, k_w, k_W, s_2 are each chosen uniformly from $\{0,1\}^\lambda$. The receiver's token $\mathcal{T}_{\mathsf{R}}$ encapsulates the code described in Figure 6, where k_s is chosen uniformly at random from $\{0,1\}^\lambda$.

Formally, each party sends the relevant code to $\mathcal{F}_{\mathsf{wrap}}$ using an appropriate (CREATE, ...) message.

Oblivious transfer phase. Following the token-exchange phase, the parties can (sequentially) run an unbounded number of sub-sessions where in each sub-session they carry out any desired number m of oblivious transfers. (Each sub-session uses only a constant number of rounds.) In each sub-session, S gets $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \langle m, \{(x_i^0, x_i^1)\}_{i=1}^m\rangle)$ and R gets $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \langle m, \{b_i\}_{i=1}^m\rangle)$ from the environment, and they execute the following protocol:

$\mathsf{S} \to \mathsf{R}$**:** For $i \in [m]$, the sender computes $a_i := \mathsf{PRF}_{k_a}(\mathsf{ssid}\|i)$, $B_i := \mathsf{PRF}_{k_B}(\mathsf{ssid}\|i)$, $w_i := \mathsf{PRF}_{k_w}(\mathsf{ssid}\|i)$, and $r_{w_i} := \mathsf{PRF}_{k_W}(\mathsf{ssid}\|i)$, and chooses $u_i \leftarrow \{0,1\}^{8\lambda^2+3\lambda}$. Here, we have $a_i \in \mathbb{F}_2^{4\lambda}$, $B_i \in \mathbb{F}_2^{4\lambda\times 4\lambda}$, and $w_i \in \{0,1\}^\lambda$. Then, the sender commits to (a_i, B_i), w_i, and

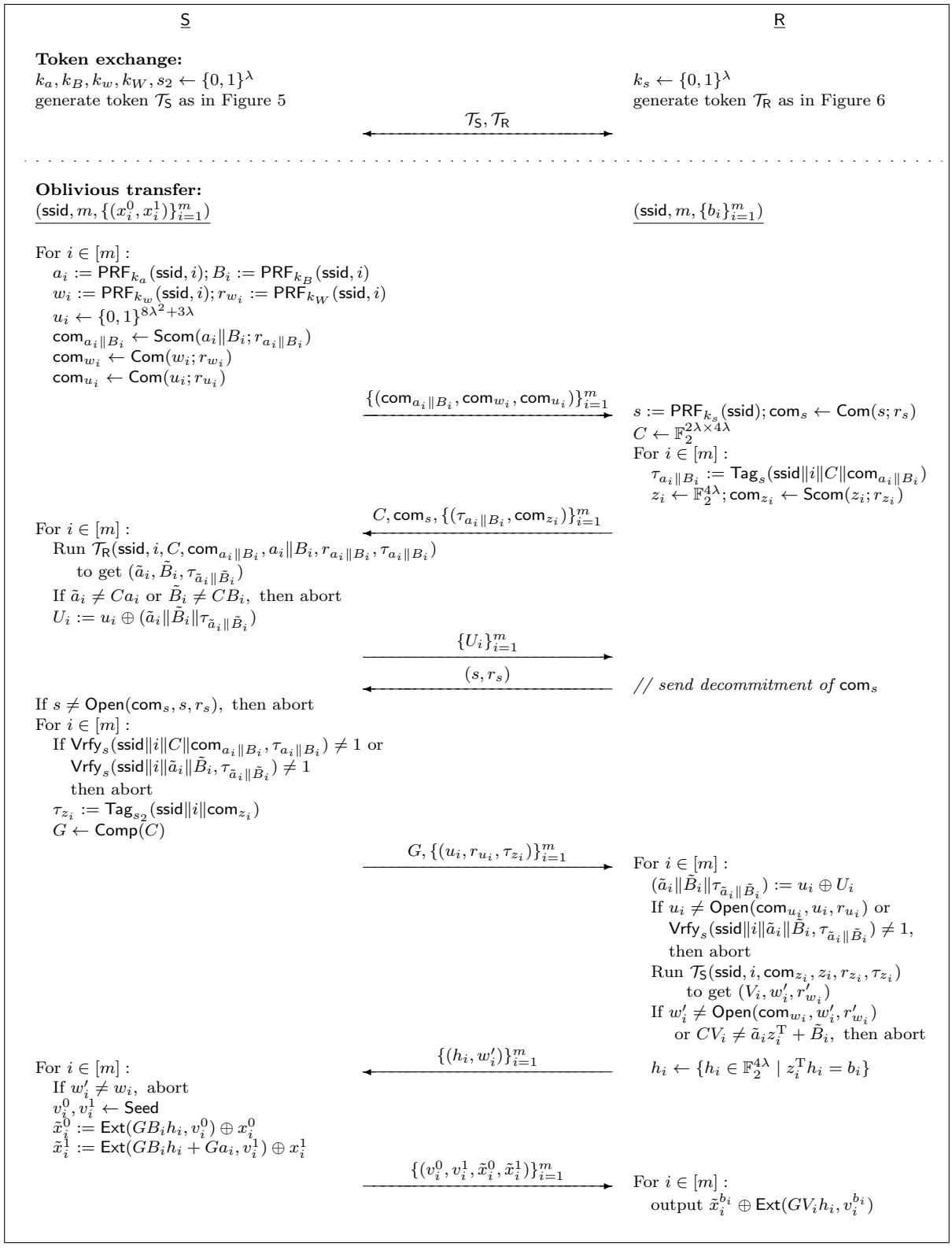

Figure 7: An OT protocol π from two stateless tokens.

u_i by executing $\mathsf{com}_{a_i\|B_i} \leftarrow \mathsf{Scom}(a_i\|B_i; r_{a_i\|B_i})$, and $\mathsf{com}_{w_i} \leftarrow \mathsf{Com}(w_i; r_{w_i})$, and $\mathsf{com}_{u_i} \leftarrow \mathsf{Com}(u_i; r_{u_i})$, respectively. The sender sends $\{(\mathsf{com}_{a_i\|B_i}, \mathsf{com}_{w_i}, \mathsf{com}_{u_i})\}_{i=1}^m$ to R.

$\mathsf{R} \to \mathsf{S}$**:** The receiver computes $s := \mathsf{PRF}_{k_s}(\mathsf{ssid})$, and commits to s by executing $\mathsf{com}_s \leftarrow \mathsf{Com}(s; r_s)$. It also chooses $C \leftarrow \mathbb{F}_2^{2\lambda\times 4\lambda}$ and $z_i \leftarrow \mathbb{F}_2^{4\lambda}$, and commits to z_i by executing $\mathsf{com}_{z_i} \leftarrow \mathsf{Scom}(z_i; r_{z_i})$. It next authenticates the commitment $\mathsf{com}_{a_i\|B_i}$ by computing

$$\tau_{a_i\|B_i} := \mathsf{Tag}_s(\mathsf{ssid}\|i\|C\|\mathsf{com}_{a_i\|B_i}) \text{ for } i = 1, \ldots, m.$$

Finally, it sends $(C, \mathsf{com}_s, \{(\tau_{a_i\|B_i}, \mathsf{com}_{z_i})\}_{i=1}^m)$ to S.

$\mathsf{S} \to \mathsf{R}$**:** For $i \in [m]$, the sender runs the token $\mathcal{T}_\mathsf{R}$ with $(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i})$ and obtains in return $(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i})$. Then S checks that $\tilde{a}_i = Ca_i$ and $\tilde{B}_i = CB_i$ for $i \in [m]$. If the check fails, S aborts the protocol. Otherwise, S computes $U_i := u_i \oplus (\tilde{a}_i\|\tilde{B}_i\|\tau_{\tilde{a}_i\|\tilde{B}_i})$ for $i \in [m]$ and sends $\{U_i\}_{i=1}^m$ to R.

$\mathsf{R} \to \mathsf{S}$**:** R sends the decommitment (s, r_s) of com_s to S.

$\mathsf{S} \to \mathsf{R}$**:** The sender validates all values transmitted so far as follows: It checks that $s = \mathsf{Open}(\mathsf{com}_s, s, r_s)$ and that $\mathsf{Vrfy}_s(\mathsf{ssid}\|i\|C\|\mathsf{com}_{a_i\|B_i}, \tau_{a_i\|B_i}) = 1$, as well as $\mathsf{Vrfy}_s(\mathsf{ssid}\|i\|\tilde{a}_i\|\tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i}) = 1$ for $i \in [m]$. If the check fails, then S aborts the protocol. Otherwise, it computes the complementary matrix $G \leftarrow \mathsf{Comp}(C)$ of C, and for $i \in [m]$ it authenticates the commitment com_{z_i} by computing $\tau_{z_i} := \mathsf{Tag}_{s_2}(\mathsf{ssid}\|i\|\mathsf{com}_{z_i})$. Finally, it sends $(G, \{(u_i, r_{u_i}, \tau_{z_i})\}_{i=1}^m)$ to R.

$\mathsf{R} \to \mathsf{S}$**:** For $i \in [m]$, the receiver checks that $u_i = \mathsf{Open}(\mathsf{com}_{u_i}, u_i, r_{u_i})$; if not, it aborts. Then R sets $\tilde{a}_i\|\tilde{B}_i\|\tau_{\tilde{a}_i\|\tilde{B}_i} := U_i \oplus u_i$. It next checks that $\mathsf{Vrfy}_s(\mathsf{ssid}\|i\|\tilde{a}_i\|\tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i}) = 1$ for $i \in [m]$. If this check does not hold, it aborts the protocol. Otherwise, for $i \in [m]$, it runs the token $\mathcal{T}_\mathsf{S}$ with $(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})$ and obtains in return (V_i, w_i', r_{w_i}'). The receiver checks that $w_i' = \mathsf{Open}(\mathsf{com}_{w_i}, w_i', r_{w_i}')$, that $CV_i = \tilde{a}_i z_i^\mathrm{T} + \tilde{B}_i$, and that $w_i' \neq \perp$ for $i \in [m]$. If the check fails, then it aborts the protocol. Otherwise, it chooses $h_i \leftarrow \mathbb{F}_2^{4\lambda}$ subject to the constraint that $b_i = z_i^\mathrm{T} h_i$ for $i \in [m]$, and sends $\{(h_i, w_i')\}_{i=1}^m$ to S.

$\mathsf{S} \to \mathsf{R}$**:** The sender checks that $w_i = w_i'$ for $i \in [m]$; if not, it aborts the protocol. Otherwise, for $i \in [m]$ it chooses $v_i^0, v_i^1 \leftarrow \mathsf{Seed}$ and computes $\tilde{x}_i^0 := \mathsf{Ext}(GB_ih_i, v_i^0) \oplus x_i^0$ and $\tilde{x}_i^1 := \mathsf{Ext}(GB_ih_i + Ga_i, v_i^1) \oplus x_i^1$. Finally it sends $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m$ to the receiver.

$\mathsf{R} \to \mathsf{S}$**:** For $i \in [m]$, the receiver computes $x_i^{b_i} := \tilde{x}_i^{b_i} \oplus \mathsf{Ext}(GV_ih_i, v_i^{b_i})$ and outputs it.

Theorem 2 *Assume* PRF *is a pseudorandom function,* Com *is computationally hiding and strongly binding,* Scom *is statistically hiding and computationally binding, and* MAC *is a deterministic, unforgeable message-authentication code. Then protocol π securely realizes $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid model.*

To show the security of the protocol, we need to construct a simulator Sim for any non-uniform PPT environment $\mathcal{Z}$ such that $\mathrm{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}},\mathsf{Sim},\mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary. Below, we briefly sketch the ideas used to construct such a simulator. Note that we assume w.l.o.g. that the adversary never asks the same query twice.

Sender Corruption. First, recall that the input of the token $\mathcal{T}_{\mathsf{R}}$ is a tuple $(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i})$ and it returns $(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i})$. Our protocol forces S^* to query $\mathcal{T}_{\mathsf{R}}$ on exactly one input (a_i, B_i) for a fixed index i. This allows Sim to extract the input from the protocol transcript. To see this, note that the receiver authenticates the inputs for its token $\mathcal{T}_{\mathsf{R}}$ and the token also authenticates its output. Thus, if the malicious sender does not query the token $\mathcal{T}_{\mathsf{R}}$ but sends a "valid" tuple $(\tilde{a}_i\|\tilde{b}_i, \tau_{\tilde{a}_i\|\tilde{b}_i})$, then this immediately contradicts the unforgeability of the MAC. Also, S^* cannot query $\mathcal{T}_{\mathsf{R}}$ with two different values $(a_i, B_i), (a'_i, B'_i)$ (for a fixed i) without contradicting the unforgeability of the tag $\tau_{a_i\|B_i}$ or the binding of $\mathsf{com}_{a_i\|B_i}$.

Next, consider the maliciously generated token $\mathcal{T}_{\mathsf{S}^*}$. The input is of a tuple $(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})$, and its output is (V_i, w'_i, r'_{w_i}). The input is chosen carefully in combination with the protocol execution such that the output of the token does not reveal any information about the choice bit $b_i = z_i^{\mathrm{T}} h_i$ of the receiver. Observe that the tag τ_{z_i} cannot contain any information about h_i since h_i has not been sent yet. Therefore, the token sees only z_i. Also, the decommitment (w'_i, r'_{w_i}) does not contain any information about z_i, since the value z_i is chosen *after* the generation of com_{w_i}.

Still, the token $\mathcal{T}_{\mathsf{S}^*}$ may send some limited amount of information about the *previous* executions to S^* by aborting the protocol. Observe that according to our definition of the functionality, aborting is only allowed once. Thus, the number ℓ of successful sub-sessions so far can encode some information of the input history of $\mathcal{T}_{\mathsf{S}^*}$. Information theoretically, the value ℓ can encode at most $\log \ell$ bits of some previous z_is. Therefore, even with this leakage, the adversary S^* has only a negligible advantage in predicting b_i. To see this, note that $b_i = z_i^{\mathrm{T}} h_i$ remains unfixed given $\log \ell$ bits of z_i.

Receiver Corruption. First, recall that the input to the token $\mathcal{T}_{\mathsf{S}}$ is $(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})$ and that its output is (V_i, w_i, r_{w_i}). As in the malicious sender case, our protocol forces the malicious receiver R^* to query the token on exactly one input z_i for each i. By observing this query, the simulator Sim can extract the input $\hat{b}_i = z_i^{\mathrm{T}} h_i$ of R^*. This property is achieved using the unforgeability of the message authentication code MAC and the binding of the commitment scheme Scom. That is, the unforgeability guarantees that the token runs only on the input (including com_{z_i}) authenticated by the honest sender, and the binding of com_{z_i} disables the malicious receiver R^* to query the token on two distinct z_is.

Next, consider the maliciously generated token $\mathcal{T}_{\mathsf{R}^*}$. The input to the token and its output are carefully handled such that it does not reveal information about (x_i^0, x_i^1) of the sender. To see this, recall the token $\mathcal{T}_{\mathsf{R}^*}$ takes as input $(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i})$ and returns $(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i})$. Now, observe the sender validates the $\tau_{\tilde{a}_i\|\tilde{B}_i}$ by checking the opening of com_s before sending $\tau_{\tilde{a}_i\|\tilde{B}_i}$. Thus, any malicious attempt by R^* and $\mathcal{T}_{\mathsf{R}^*}$ to gain information about (a_i, B_i) beyond $(\mathsf{com}_{a_i\|B_i}, \tilde{a}_i, \tilde{B}_i)$ (e.g., by establishing a covert channel using $\tau_{\tilde{a}_i\|\tilde{B}_i}$) will be caught.

Still, the token $\mathcal{T}_{\mathsf{R}^*}$ may send some limited amount of information about the previous executions to R^* by aborting the protocol (only allowed one-time), that is, the number ℓ of successful sub-sessions so far can encode some information of the input history of $\mathcal{T}_{\mathsf{R}^*}$. Note that this value ℓ can encode at most $\log \ell$ bits. Using appropriate extractors in computing the masks for $\tilde{x}_i^0$ and $\tilde{x}_i^1$, we can handle this amount of leakage.

4.2 Corrupted Sender

Lemma 1 *Assume* PRF *is a pseudorandom function,* Com *is computationally hiding and strongly binding,* Scom *is statistically hiding and computationally binding, and* MAC *is a deterministic, unforgeable message-authentication code. Then the construction defined in Figure 7 is secure against a malicious sender.*

Here we consider the case that the sender is corrupted by the dummy adversary while the receiver is honest. We present the following ideal-world simulator Sim such that for any environment $\mathcal{Z}$ it holds that $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\pi,\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}},\mathsf{Sim},\mathcal{Z}}$. The simulator Sim in the ideal world internally simulates a dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$. To do so, the simulator forwards each message from/to the environment $\mathcal{Z}$ to/from the internal adversary $\mathcal{A}$. In addition, Sim simulates the following interactions with the internal attacker $\mathcal{A}$.

Simulation of the initial token exchanges:

1. The simulator Sim simulates $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ honestly and returns $(\text{READY}, \langle \mathsf{sid}, \mathsf{R}, \mathsf{S}\rangle)$ to $\mathcal{A}$.[4]
2. Upon receiving $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathcal{M}_{\mathcal{A}})$ from $\mathcal{A}$ to the simulated $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$, the simulator records $\mathcal{M}_{\mathcal{A}}$.

Simulation of the execution for sub-session ssid**:** The algorithm Sim simulates the protocol execution as follows: It chooses a vector of bits $\hat{b}_1, \ldots, \hat{b}_m$ with $\hat{b}_i = 0$ for $i \in [m]$, and it simulates $\mathsf{R}(\mathsf{ssid}, m, \{\hat{b}_i\}_{i=1}^m)$ honestly, except for the following modifications:

1. When the protocol specification of the honest receiver requires invocation of $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$, the simulator Sim handles it by running the code $\mathcal{M}_{\mathcal{A}}$ directly as $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ would.
2. When $\mathcal{A}$ queries $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ on a tuple $(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i})$, the simulator Sim simulates $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ honestly throughout the execution with the only exception that it uses a random function F instead of PRF. This modification changes the success probability of the adversary only by a negligible amount. It additionally records the pair (a_i, B_i) if the query is valid (i.e., C is the value that is indeed chosen by Sim and the output is not $(\perp, \perp, \perp)$).
3. If $\mathcal{A}$ does not make exactly one valid query per (ssid, i) to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$, output abort and terminate.
4. Consider the move in the protocol where $\mathcal{A}$ sends $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m$. Let $\{(V_i, w_i', r'_{w_i})\}_{i=1}^m$ be the output by $\mathcal{M}_{\mathcal{A}}$ with $\{(\mathsf{ssid}, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})\}_{i=1}^m$ given as input. Choose z_i' at random such that $z_i'^{\mathrm{T}} h_i = \hat{b}_i \oplus 1$. Compute $V_i' := a_i z_i'^{\mathrm{T}} + B_i$. Compute $x_i^{\hat{b}_i} := \tilde{x}_i^{\hat{b}_i} \oplus \mathsf{Ext}(GV_i h_i, v_i^{\hat{b}_i})$ and $x_i^{\hat{b}_i \oplus 1} := \tilde{x}_i^{\hat{b}_i \oplus 1} \oplus \mathsf{Ext}(GV_i' h_i, v_i^{\hat{b}_i \oplus 1})$. Then the simulator Sim in the name of the corrupted sender sends $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \langle m, \{(x_i^0, x_i^1)\}_{i=1}^m\rangle)$ to the functionality $\mathcal{F}_{\mathsf{multi\text{-}OT}}$.

We show computational indistinguishability between the real and ideal worlds by the following hybrids.

Hybrid $\mathcal{H}_0$: This is the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world execution $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\mathcal{A},\pi,\mathcal{Z}}$.

[4] For readability, $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ (resp., $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$) denotes the $\mathcal{F}_{\mathsf{wrap}}$ functionality that manages the token $\mathcal{T}_{\mathsf{S}}$ (resp., $\mathcal{T}_{\mathsf{R}}$).

Hybrid $\mathcal{H}_1$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_1,\mathcal{Z}}$. The ideal functionality $\mathcal{F}$ is the same as $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ except that it additionally provides the simulator with the real inputs $\{b_i\}_{i=1}^m$ of the receiver. The simulator Sim_1 in the ideal world internally simulates the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$. That is, the simulator forwards each message from/to the environment $\mathcal{Z}$ to/from the internal adversary $\mathcal{A}$. In addition, Sim_1 simulates the following interactions with the internal attacker $\mathcal{A}$, playing as the receiver following the protocol description honestly by using the given input $\{b_i\}_{i=1}^m$ from $\mathcal{F}$. Moreover, Sim_1 simulates $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ and $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ honestly. Note that the queries to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ from $\mathcal{A}$ can be observed by the simulator Sim. Once finishing the protocol execution, Sim_1 will obtain $\{x_i^{b_i}\}_{i=1}^m$ from the protocol execution with $\mathcal{A}$. Then, Sim_1 sets $x_i^{1-b_i} := 0^\lambda$ for $i \in [m]$ and sends $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \langle m, \{(x_i^0, x_i^1)\}_{i=1}^m\rangle)$ to the functionality $\mathcal{F}$. Note that $\mathcal{F}$ will pass only $x_i^{b_i}$ to the dummy receiver in the ideal world.

ANALYSIS: Obviously, $\mathcal{H}_0$ and $\mathcal{H}_1$ are identically distributed.

Hybrid $\mathcal{H}_2$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_2,\mathcal{Z}}$. The simulator Sim_2 is the same as Sim_1, except that it uses a truly random function F instead of a pseudorandom function when simulating $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$.

ANALYSIS: From the properties of pseudo-random functions, $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable.

Hybrid $\mathcal{H}_3$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_3,\mathcal{Z}}$. The simulator Sim_3 is the same as Sim_2, except the following:

- Let $\tau_{a_i\|B_i}$ be the MAC tag that Sim_3 generated and sent to $\mathcal{A}$. This tag is a MAC on the message $(\mathsf{ssid}\|i\|C\|\mathsf{com}_{a_i\|B_i})$, for the OT execution with respect to (ssid, i). The simulator Sim_3 outputs abort if $\mathcal{A}$ makes any query $(\mathsf{ssid}, i, C', \mathsf{com}'_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau'_{a_i\|B_i})$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ such that

$$(C', \mathsf{com}'_{a_i\|B_i}, \tau'_{a_i\|B_i}) \neq (C, \mathsf{com}_{a_i\|B_i}, \tau_{a_i\|B_i}),$$

  but the tag $\tau'_{a_i\|B_i}$ is valid, i.e., $\mathsf{Vrfy}_s(\mathsf{ssid}\|i\|C\|\mathsf{com}'_{a_i\|B_i}, \tau'_{a_i\|B_i}) = 1$ for some (ssid, i).

ANALYSIS: We show that the probability that Sim_3 outputs abort is negligible in λ. This follows from the unforgeability of the deterministic, canonical MAC and the hiding property of com_s. Observe that since MAC is deterministic, it must be the case that $(C, \mathsf{com}_{a_i\|B_i}) \neq (C', \mathsf{com}'_{a_i\|B_i})$. We refer to this event as forge. Therefore, it suffices to show $\Pr[\mathsf{forge}]$ is negligible in λ.

Towards contradiction, suppose that there is an environment $\mathcal{Z}^*$ and the dummy adversary $\mathcal{A}$ that cause forge to occur with non-negligible probability. W.l.o.g., we assume $\mathcal{Z}^*$ to be deterministic with randomness encoded in the advice. By an averaging argument, there is some fixed ssid^* where $\mathcal{Z}^*$ makes this event occur with non-negligible probability. Hereafter, this ssid^* will be treated as an implicit advice to any non-uniform algorithm. Consider the following partial hybrid:

Sub-Hybrid $\mathcal{H}_{3a}$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_{3a},\mathcal{Z}^*}$. The simulator Sim_{3a} the same as Sim_3 except that in sub-session ssid^* it sets $\mathsf{com}_s \leftarrow \mathsf{Com}(0^\lambda)$ and it terminates immediately after receiving the U_i's in sub-session ssid^*.

From the computational hiding property of com_s, the probability that the event forge occurs in sub-session ssid^* in $\mathcal{H}_{3a}$ is still non-negligible; otherwise, we can construct a non-uniform adversary $\mathcal{B}$ breaking the hiding property of com_s as follows.

> $\mathcal{B}$ is given ssid^* as non-uniform advice. Then $\mathcal{B}$ chooses two messages (m_0, m_1) where, $m_0 = 0^\lambda$ and a m_1 is a random string s. $\mathcal{B}$ receives com from outside (a commitment to 0^λ or s) and runs $\mathcal{H}_{3a}$ until it receives the U_i's in the sub-session ssid^* using the received com as com_s in sub-session ssid^*. $\mathcal{B}$ outputs 1 if forge occurs and 0 otherwise.

Clearly, if com is a commitment to s then this is the same as $\mathcal{H}_3$ up to the point that the U_i's are received and so, by assumption, the probability of forge is non-negligible. Thus, the probability of forge must also be non-negligible when com is a commitment to 0^λ as otherwise $\mathcal{B}$ breaks the hiding of com.

To complete the proof we now show that, by the unforgeability of MAC, the probability of forge is negligible when com is a commitment to 0^λ. Assume that this is not the case, the following adversary $\mathcal{B}'$ breaks the unforgeability of MAC:

> $\mathcal{B}'$ runs $\mathcal{H}_{3a}$ while answering the token queries from $\mathcal{Z}^*$ in sub-session ssid^* by obtaining the tags for the token inputs from the outside MAC tagging oracle. Then, upon seeing any token query $(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau'_{a_i\|B_i})$ from $\mathcal{Z}^*$ with a tag not coming from the tagging algorithm, $\mathcal{B}'$ returns the forgery (m, τ) as $(\mathsf{ssid}\|i\|C\|\mathsf{com}_{a_i\|B_i}, \tau'_{a_i\|B_i})$. Since com_s is independent of the MAC key s, this is indeed a forgery of MAC and thus can only happen with negligible probability.

Hybrid $\mathcal{H}_{3b}$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_{3b},\mathcal{Z}}$. The simulator Sim_{3b} is the same as Sim_3, except for the following:

- The simulator Sim_{3b} outputs abort if $\mathcal{A}$ makes any two valid queries

  $$(\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i}) \quad \text{and} \quad (\mathsf{ssid}, i, C, \mathsf{com}_{a_i\|B_i}, a'_i, B'_i, r'_{a_i\|B_i}, \tau_{a_i\|B_i})$$

  to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ such that

  $$(a_i, B_i) \neq (a'_i, B'_i).$$

ANALYSIS: The above case happens if the commitments are equal, but the decommitments differ. The case when the commitments are different is already handled in $\mathcal{H}_3$. Assume that Sim_{3b} outputs abort with non-negligible probability. It is easy to see that this yields two different decommitments for $\mathsf{com}_{a_i\|B_i}$ with non-negligible probability. However, this can be used to break the binding of $\mathsf{com}_{a_i\|B_i}$ giving a contradiction.

Hybrid $\mathcal{H}_4$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_4,\mathcal{Z}}$. The simulator Sim_4 is the same as Sim_{3b}, except that Sim_4 outputs abort if $\mathcal{A}$ makes no valid queries with some (ssid, i) to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$, while sending a valid tag $\tau_{\tilde{a}_i\|\tilde{B}_i}$ (i.e., $u_i \oplus U_i = \tilde{a}_i\|\tilde{B}_i\|\tau_{\tilde{a}_i\|\tilde{B}_i}$).

ANALYSIS: Intuitively, the strong binding of com_{u_i} guarantees that the malicious sender is committed to u_i before seeing C and the MAC key s. Thus, in order to send a "valid" U_i, it either has to break the binding of the commitment scheme, or it has to forge the MAC. Note

that since com_{u_i} is strongly-binding the probability that it can be broken is negligible. One issue here is that the adversary sends the decommitment (u_i, r_{u_i}) *after* seeing the key of the MAC s. We handle this by exploiting the non-uniformity of the MAC. This property allows us to perform an *unbounded* computation *before* getting access to the tagging oracle (since the computation is captured by the auxiliary advice). Once it gets access to the tagging algorithm, the forger runs in polynomial time. We will use this unbounded step to break the hiding of com_{u_i} and recover the decommitment (u_i, r_{u_i}) and give it as non-uniform input to the adversary. Note that since com_{u_i} is strongly-binding we find the correct decommitment with overwhelming probability.

Claim 1 *If* Com *is a strongly-binding commitment scheme and* MAC *a non-uniform unforgeable message authentication code, the probability that* Sim_4 *aborts is negligible in* λ.

To prove the claim, suppose that there is an environment $\mathcal{Z}^*$ and the dummy adversary $\mathcal{A}$ that cause Sim_4 to abort with non-negligible probability. W.l.o.g., we assume $\mathcal{Z}^*$ to be deterministic with randomness encoded in the advice. By an averaging argument, there is some fixed ssid^* where $\mathcal{Z}^*$ makes Sim_4 abort with non-negligible probability. Hereafter, this ssid^* will be treated as an implicit advice to any non-uniform algorithm. Consider the following partial hybrid:

Sub-Hybrid $\mathcal{H}_{4a}$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_{4a},\mathcal{Z}^*}$. The simulator Sim_{4a} the same as Sim_4 except that it sets $\mathsf{com}_s \leftarrow \mathsf{Com}(0^\lambda)$ in sub-session ssid^* and terminates the whole execution right before sending the decommitment of com_s in sub-session ssid^*

Just as in the analysis of hybrid $\mathcal{H}_{3a}$, the probability that Sim aborts in $\mathcal{H}_{4a}$ must be (essentially) the same as the probability that it aborts in $\mathcal{H}_4$, as otherwise we can construct a non-uniform algorithm (taking ssid^* and (u_i, r_{u_i})s as advice) that breaks the hiding of com_s. Thus, by assumption, this probability is non-negligible.

We can now conclude the proof by demonstrating that, by the non-uniform unforgeability of MAC, the probability that Sim aborts in $\mathcal{H}_{4a}$ must be negligible, giving a contradiction. Assume that this is not the case. Then, we demonstrate the following adversary $\mathcal{B}$ breaking the unforgeability of MAC:

> First, $\mathcal{B}$ extracts $\{(u_i, r_{u_i})\}_{i=1}^m$ that $\mathcal{Z}^*$ uses in sub-session ssid^*. Now, $\mathcal{B}$ runs $\mathcal{H}_{4a}$ while answering the token queries from $\mathcal{Z}^*$ in sub-session ssid^* by obtaining the tags $\tau_{a_i \| B_i}$ from the outside MAC tagging oracle. Then, $\mathcal{B}$ computes $\tilde{a}_i \| \tilde{B}_i \| \tau_{\tilde{a}_i \| \tilde{B}_i} := u_i \oplus U_i$ for any i^{th} OT for which $\mathcal{Z}^*$ did not make a query to the token. Finally, $\mathcal{B}$ outputs the forgery (m, τ) as $((\mathsf{ssid}^* \| i \| \tilde{a}_i \| \tilde{B}_i), \tau_{\tilde{a}_i \| \tilde{B}_i})$.

Hybrid $\mathcal{H}_5$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_5,\mathcal{Z}}$. The simulator Sim_5 is the same as $\mathcal{H}_4$, except that $\mathcal{H}_5$ outputs abort if $\mathcal{M}_\mathcal{A}$ outputs V such that $V \neq az^{\mathrm{T}} + B$ and $CV = \tilde{a}z^{\mathrm{T}} + \tilde{B}$.

ANALYSIS: Let $V' = az^{\mathrm{T}} + B$, then, if $\mathcal{M}_\mathcal{A}$ outputs $V \neq V'$ such that $CV = CV' = \tilde{a}z^{\mathrm{T}} + \tilde{B}$ then we know that $CVh = CV'h$. However, this implies that we have found a collision in the function specified by the random matrix C, which is not possible except with negligible probability when C is unknown. Thus, $\mathcal{H}_5$ will output abort with probability negligible in λ and thus, $\mathcal{H}_4 \approx H_5$.

Hybrid $\mathcal{H}_6$: This is $\text{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}},\mathsf{Sim}_6,\mathcal{Z}}$. The simulator Sim_6 is the same as Sim_5 except the following:

- Sim_6 chooses z_i such that $z_i^T h_i = 0$ and follows the protocol specification. Let $(V_i, w_i', r_{w_i'})$ be the output from $\mathcal{M}_\mathcal{A}$.
- Sim_6 also chooses z_i' at random such that $z_i'^{\mathrm{T}} h_i = 1$ and computes $V_i' := a_i z_i'^{\mathrm{T}} + B_i$. Then, it computes $x_i^0 := \tilde{x}_i^0 \oplus \mathsf{Ext}(GV_{b_i} h_i, v_i^0)$ and $x_i^1 := \tilde{x}_i^1 \oplus \mathsf{Ext}(GV' h_i, v_i^1)$. Finally, the simulator in the name of the corrupted sender sends $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \langle m, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle)$ to $\mathcal{F}_{\mathsf{multi\text{-}OT}}$.

We note that Sim_6 is the simulator Sim described before.

ANALYSIS: We show that these both hybrids are computational indistinguishable for any $\mathcal{Z}$. To do so, assume that ℓ sub-sessions have successfully completed for some $\ell \leq poly(\lambda)$. We now argue that the views of $\mathcal{Z}$ in $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable. We prove this via induction on ℓ. Obviously, the case $\ell = 0$ follows easily because both views are empty. Now we would like to show that $\mathcal{H}_5 \approx \mathcal{H}_6$ after $\ell+1$ sub-sessions. To do so, we make the inductive hypothesis that the views are close after ℓ subsessions. First, consider the case that the $(\ell+1)^{th}$ session does not abort.

- If $b_i = 0$, the value x_i^0 is computed exactly as the honest receiver does in the real execution. Thus, the true value is learned.
- If $b_i = 1$, the value z_i' is also uniformly distributed to the view of $\mathcal{M}_\mathcal{A}$ (oblivious of h_i), the value V_i' (as it would be output by $\mathcal{M}_\mathcal{A}$) must be distributed identically to V_i in $\mathcal{H}_5$. Thus, setting $V_i' := a_i z_i'^{\mathrm{T}} + B_i$ leads to correct decryption of x_i^1.

Now, consider the case where $\mathcal{A}$ and $\mathcal{Z}$ cause the $(\ell+1)^{th}$ sub-session to abort, eventually based on the maliciously generated token. Observe that up to this point $\mathcal{H}_5 \approx \mathcal{H}_6$ and thus, the abort happens in both hybrids with the same probability (except for a negligible amount). In the following we bound the information that the token can transport to $\mathcal{Z}$ by causing an abort in the protocol. In particular, the token may try to encode some information about the previous z_is. Since the commitment com_{w_i} is strongly binding, it follows that even an unbounded adversary cannot find distinct opening. Thus, w_i' cannot contain any information about z_i. Information theoretically, however, the value ℓ can encode at most $\log \ell$ bits of some previous z_is. Recall that b_i is the inner product of z_i^{T} and h_i. Therefore, even with this leakage, the adversary S^* has only a negligible advantage in predicting b_i. Thus, $\mathcal{H}_5 \approx \mathcal{H}_6$.

4.3 Corrupted Receiver

Lemma 2 *Assume* PRF *is a pseudorandom function,* Com *is computationally hiding and strongly binding,* Scom *is statistically hiding and computationally binding, and* MAC *is a deterministic, unforgeable message-authentication code. Then the construction defined in Figure 7 is secure against a malicious receiver.*

Let $\mathcal{A}$ be the dummy adversary corrupting the receiver. We present the ideal-world simulator Sim such that for any environment $\mathcal{Z}$ it holds that $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\pi,\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}},\mathsf{Sim},\mathcal{Z}}$. The simulator Sim internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the

external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. In addition, Sim simulates the following interactions with the internal $\mathcal{A}$.

Initial Token Exchanges:

1. Simulates a copy of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ and return $(\text{READY}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to $\mathcal{A}$.
2. Upon receiving $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{R}, \mathsf{S} \rangle, \mathcal{M}_{\mathcal{A}})$ from $\mathcal{A}$ to the simulated $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$, store $\mathcal{M}_{\mathcal{A}}$.

OT Execution with ssid:

1. Simulate $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ honestly throughout the execution with only exception that PRF is replaced with a purely random function. Also, simulate $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ honestly throughout the execution by running $\mathcal{M}_{\mathcal{A}}$ directly.
2. Follow the protocol honestly, playing as the sender up to the point where $\mathcal{A}$ has sent $\{(h_i, w_i')\}_{i=1}^m$.
3. If $\mathcal{A}$ didn't make exactly one valid query per (ssid, i) to the simulated $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$, abort and terminate. Otherwise, set $\hat{b}_i := z_i^{\mathrm{T}} h_i$, where z_i is the opening of com_{z_i} extracted from the token query, and query $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ with $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \langle m, \{\hat{b}_i\}_{i=1}^m \rangle)$ and obtains $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{x_i^{\hat{b}_i}\}_{i=1}^m)$.
4. Send $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m$ to $\mathcal{A}$ according to the protocol description, except that $x_i^{1-\hat{b}_i}$s are chosen at random.

We show computational indistinguishability between the real and ideal worlds by the following hybrids.

Hybrid $\mathcal{H}_0$: This is the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world execution $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{wrap}}}$.

Hybrid $\mathcal{H}_1$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_1,\mathcal{Z}}$. The functionality $\mathcal{F}$ is the same as $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ except that it additionally provides the simulator with the inputs $(\mathsf{ssid}, m, \{(x_i^0, x_i^1)\}_{i=1}^m)$ of the sender. The simulator Sim_1 works as follows:

1. The simulator Sim_1 internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. In addition, Sim_1 simulates the following interactions with the internal $\mathcal{A}$.
2. Play as the sender honestly in the execution with $\mathcal{A}$, using the input of the sender coming from $\mathcal{F}$.
3. Simulate $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ honestly. Also, simulate $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ honestly throughout the execution by running $\mathcal{M}_{\mathcal{A}}$ directly.

ANALYSIS: Obviously, $\mathcal{H}_0$ and $\mathcal{H}_1$ are identically distributed.

Hybrid $\mathcal{H}_2$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_2,\mathcal{Z}}$. The simulator Sim_2 works the same as Sim_1, except that PRF is replaced by a purely random function when simulating $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$.

ANALYSIS: From the properties of pseudo-random functions, $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable.

Hybrid $\mathcal{H}_3$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_3,\mathcal{Z}}$. The simulator Sim_3 works the same as Sim_2, except that Sim_3 outputs abort if there exists an $i \in \{1,\ldots,m\}$ such that $\mathcal{A}$ sends the correct w'_i while making zero query to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ with index i in some sub-session ssid.

ANALYSIS: We show that the probability that Sim_3 outputs abort is negligible in λ. Roughly speaking, from the hiding property of com_{w_i}, this probability is negligible. In particular, given the dummy adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ that makes Sim_3 output abort with non-negligible probability, one can easily construct a non-uniform adversary $\mathcal{B}$ that breaks the hiding property of the commitment scheme as follows:

> Let i^*, ssid^* be the index i and the sub-session in which Sim_3 aborts with non-negligible probability (such an ssid^* and i^* must exist by an averaging argument). $\mathcal{B}$ is given i^* and ssid^* as non-uniform advice and chooses two messages (m_0, m_1) to break the hiding property, where m_0 is 0^λ and m_1 is w_{i^*}. Remember that this w_{i^*} is chosen uniformly at random from $\{0,1\}^\lambda$. Given the commitment com to m_0 or m_1, $\mathcal{B}$ works as Sim_3 using com as $\mathsf{com}_{w_{i^*}}$ in the OT indexed with (ssid^*, i^*). If Sim_3 aborts because of $\mathcal{A}$ making no valid query (but outputting the correct w_{i^*}) on this index, then $\mathcal{B}$ outputs 1; otherwise $\mathcal{B}$ outputs 0.

If com is a commitment to m_0, the probability that $\mathcal{B}$ outputs 1 is exactly $2^{-\lambda}$, since w_{i^*} is chosen randomly and independently of the view of $\mathcal{A}$ and $\mathcal{Z}$. On the other hand, if com is a commitment to m_1, $\mathcal{B}$ outputs 1 with non-negligible probability from the assumption about $\mathcal{A}$ and $\mathcal{Z}$. Hence, $\mathcal{B}$ has a non-negligible advantage in breaking the hiding property of the commitment.

Hybrid $\mathcal{H}_4$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_4,\mathcal{Z}}$. The simulator Sim_4 works the same as Sim_3, except that Sim_4 outputs abort if $\mathcal{A}$ sends a correct w'_i, but makes at least two valid, distinct queries

$$(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i}) \neq (\mathsf{ssid}, i, \mathsf{com}'_{z_i}, z'_i, r'_{z_i}, \tau'_{z_i})$$

in some (ssid, i) to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ such that $z_i \neq z'_i$.

ANALYSIS: We show that the probability that Sim_4 outputs abort is negligible in λ.

Note that this event occurs when $\mathcal{A}$ provides two different decommitments for com_{z_i} (event bind) or when $\mathcal{A}$ makes a valid query with com'_{z_i} different from com_{z_i} (event forge). The former case directly violates the binding property of the commitment scheme and thus, $\Pr[\mathsf{bind}]$ is negligible in λ. The latter case directly breaks the unforgeability of MAC and thus, $\Pr[\mathsf{forge}]$ is also negligible in λ. Since both events happen only with negligible probability, it follows that the hybrids are indistinguishable.

Hybrid $\mathcal{H}_5$: This is $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_5,\mathcal{Z}}$. The simulator Sim_5 works the same as Sim_4 except for the following:

- The simulator Sim_5 runs in exponential time in λ, and it extracts the key s for MAC contained in com_s for each sub-session ssid.
- In some sub-session ssid, if the token output is inconsistent, i.e., $\mathsf{Vrfy}_s(\mathsf{ssid}\|i\|\tilde{a}_i\|\tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i}) \neq 1$, then Sim_4 chooses U_i at random and sends it, instead of computing $U_i := u_i \oplus (\tilde{a}_i\|\tilde{B}_i\|\tau_{\tilde{a}_i\|\tilde{B}_i})$.

ANALYSIS: We show the indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_5$. Roughly speaking, the strong binding property of com_s guarantees that the extractor obtains s, moreover, from the hiding property of com_{u_i}, the two hybrids are indistinguishable. First note that if the output of the token is inconsistent in some sub-session then, in both $\mathcal{H}_4$ and $\mathcal{H}_5$, S will abort after receiving the decommitment of com_s, so we only need to argue that the views are indistinguishable up to this point. If the token output is never inconsistent then $\mathcal{H}_4$ and $\mathcal{H}_5$ are identical. Thus, the only difference between these two hybrids is that in the sub-session in which the token output is inconsistent Sim_4 still sends the correct U_i, while Sim_5 sends a random U_i' instead. Now suppose that $\mathcal{A}$ is a dummy adversary and that there exists an environment $\mathcal{Z}$ that distinguishes the two distributions with non-negligible probability. Then, by an averaging argument, there exists an environment $\mathcal{Z}^*$ that distinguishes them while always using some fixed key s^* for MAC and in some fixed sub-session ssid^*. By the above discussion, the protocol aborts in this sub-session.

In order to see this, we define the following intermediary hybrid distributions. For $j \in \{0, \ldots, m\}$, let $\mathcal{H}_{4,j}$ be $\text{IDEAL}_{\mathcal{F},\mathsf{Sim}_{4,j},\mathcal{Z}^*}$ where $\mathsf{Sim}_{4,j}$ runs Sim_5 for the first j OT's in sub-session ssid^* and then runs Sim_4 for the remaining OT's in this sub-session. Note that $\mathcal{H}_{4,0} = \mathcal{H}_4$ and $\mathcal{H}_{4,m} = \mathcal{H}_5$. Therefore, the environment $\mathcal{Z}^*$ must distinguish $\mathcal{H}_{4,j^*-1}$ and $\mathcal{H}_{4,j^*}$ for some $j^* \in [m]$ with non-negligible probability. Given s^*, ssid^* and j^*, we show a non-uniform polynomial time algorithm $\mathcal{B}$ that breaks the computational hiding property of the commitment scheme.

> $\mathcal{B}$ chooses two messages (m_0, m_1) to break the hiding property: m_0 is u_{j^*} and m_1 is an independently chosen random u'_{j^*}. Given the commitment com to m_0 or m_1, $\mathcal{B}$ runs a protocol with $\mathcal{A}$ and $\mathcal{Z}^*$ while working as Sim_{4,j^*-1} using com as $\mathsf{com}_{u_{j^*}}$ in sub-session ssid^*. After the protocol aborts in sub-session ssid^*, $\mathcal{B}$ outputs whatever $\mathcal{Z}^*$ outputs.

Note that, if com is a commitment to $m_0 = u_{j^*}$ then on index j, this is exactly the distribution output by Sim_4 and so the full view of $\mathcal{A}$ and $\mathcal{Z}^*$ is identical to their view in hybrid $\mathcal{H}_{4,j^*-1}$ (since there are only $j^* - 1$ execution of Sim_5). On the other hand, if com is a commitment to $m_1 = u'_{j^*}$ then this is exactly the hybrid $\mathcal{H}_{4,j^*}$ (since u'_{j^*} is independent of u_{j^*}). Thus, by our assumption on $\mathcal{Z}^*$, $\mathcal{B}$ can break the hiding of com with non-negligible probability.

Hybrid $\mathcal{H}_6$: This is $\text{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}},\mathsf{Sim}_6,\mathcal{Z}}$. The simulator Sim_6 works the same as Sim_5, except that Sim_6 computes $\hat{b}_i = z^{\mathrm{T}}h$, where z is extracted from the token query, and obtains $x_i^{\hat{b}_i}$ from $\mathcal{F}_{\mathsf{multi\text{-}OT}}$ but chooses $\tilde{x}_i^{1-\hat{b}_i}$ at random in each OT execution indexed with (ssid, i).

ANALYSIS: We show that these hybrids are statistically close. Suppose that ℓ sub-sessions have successfully completed for some $\ell \leq poly(\lambda)$.

We first argue that the views of $\mathcal{Z}$ in $\mathcal{H}_6$ and $\mathcal{H}_5$ are statistically close. We do this by induction on ℓ. The base case with $\ell = 0$ is trivial as both views are empty. Now, assuming that the distributions are statistically close after $\ell - 1$ sub-sessions, we need to show that they are still close after ℓ sub-sessions. First observe the following,

- Since, by assumption, all the execution completed successfully, the output of the token was always consistent. In particular, this means that this output does not reveal any information to $\mathcal{A}$ and $\mathcal{Z}$ as it is completely determined by the token's input.

- Since commitment $\mathsf{com}_{a_i \| B_i}$ is statistically-hiding, it is statistically close to a commitment $\mathsf{com}_{a_i^*, B_i^*}$ for any a^*, B^*.

Now, we consider the view of $\mathcal{A}$ and $\mathcal{Z}$ in the ℓth sub-session. In particular, we split our analysis into two cases: $\hat{b}_i = 0$ and $\hat{b}_i = 1$. In the first case (i.e., $\hat{b}_i = 0$), note that given $(C, G, \tilde{a}_i, \tilde{B}_i, z_i, h_i, V_i)$ with $z_i^{\mathrm{T}} h_i = 0$ and $V_i = \tilde{a}_i z_i^{\mathrm{T}} + \tilde{B}_i$, for any $X \in \mathbb{F}_2^{2\lambda}$, there always exists a pair (a_i^*, B_i^*) such that

$$\tilde{a}_i = Ca_i^*, \tilde{B}_i = CB_i^*, GV_i h_i = GB_i^* h_i, X = GB_i^* h_i + Ga_i^*.$$

This stems from the fact that C and G are complimentary matrices. Next, consider the case when $\hat{b}_i = 1$. Similarly, given $(C, G, \tilde{a}_i, \tilde{B}_i, z_i, h_i, V_i)$ with $z_i^{\mathrm{T}} h_i = 1$ and $V_i = \tilde{a}_i z_i^{\mathrm{T}} + \tilde{B}_i$, for any $X \in \mathbb{F}_2^{2\lambda}$, there always eists a pair (a_i^*, B_i^*) such that

$$\tilde{a}_i = Ca_i^*, \tilde{B}_i = CB_i^*, X = GB_i^* h_i, GV_i h_i = GB_i^* h_i + Ga_i^*.$$

Thus, since the distribution of com_{a_i, B_i} is statistically close to that of $\mathsf{com}_{a_i^*, B_i^*}$ for any a^*, B^*, the distribution of X to the view of $\mathcal{Z}$, and thus also $\tilde{x}_i^{1-\hat{b}_i}$ (since X is used to mask this value) is statistically close to uniform.

Now consider the case that $\mathcal{A}$ and $\mathcal{Z}$ (possibly using the malicious token generated by them) cause sub-session $\ell + 1$ to abort. First, since up to this point the distributions of $\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically close, the probability that an abort happens is the same (except with negligible difference) in the two hybrids. We now need to bound the amount of information conveyed from the token to $\mathcal{Z}$ by this abort. The token may try to choose the number $\ell + 1$ of sub-session to abort in in order to encode some information about its input in one of the previous sub-sessions to help $\mathcal{Z}$ distinguish between the two hybrids. Specifically, the token may try to leak some bits of the pad X used to mask $x_i^{1-\hat{b}_i}$ for some ssid and some i. However, since $\ell \leq poly(\lambda)$, the value of ℓ can only reveal at most $O(\log \lambda)$ bits. Thus, the pad X would still have $2\lambda - O(\log \lambda)$ bits of min-entropy and so, by using a strong extractor Ext, we get a pad that is statistically close to uniform, thus statistically hiding the value of $x_i^{1-\hat{b}_i}$ and making the hybrids statistically close.

Hybrid $\mathcal{H}_7$: This is $\text{IDEAL}_{\mathcal{F}_{\mathsf{multi\text{-}OT}}, \mathsf{Sim}_7, \mathcal{Z}}$. The simulator Sim_7 works the same as Sim_6 except the following:

- Sim_7 runs in time polynomial in λ. Now, U_i is always computed as $u_i \oplus (\tilde{a}_i \| \tilde{B}_i \| \tau_{\tilde{a}_i \| \tilde{B}_i})$.

We note that Sim_7 is the simulator Sim described before.

ANALYSIS: The indistinguishability between $\mathcal{H}_6$ and $\mathcal{H}_7$ can be similarly shown to that between $\mathcal{H}_4$ and $\mathcal{H}_5$.

5 Black-Box Impossibility of OT Using One Stateless Token

In the previous section we showed that an unbounded number of universally composable OTs (and hence universally composable secure computation) is possible by having the parties exchange two stateless tokens. We show here that a construction of (even a single) OT using *one* stateless token

is impossible using "black-box techniques" alone. Here, by "black-box" we refer to the simulator Sim's access to the code $\mathcal{M}$ that is encapsulated in the tokens. This may seem a bit strange since the token model as defined by Katz [Kat07] is inherently nonblack-box and, in particular, the simulator is given the code $\mathcal{M}$ that the malicious party submits to $\mathcal{F}_{\mathsf{wrap}}$. Nevertheless, an examination of the proof of Theorem 2 — as well as the proofs of security in all prior work [CGS08, MS08, GKR08, DNW09, GIMS10, GIS$^+$10, DKM11] — shows that the simulator only uses this code in a black-box fashion, namely, by running the code to observe its input/output behavior but without using any internal structure of the code itself. We show that, restricting to simulators of this type, constructions of OT from one stateless token are impossible.

Theorem 3 *There is no protocol π that uses one stateless token to securely realize $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid model whose security is proven using a simulator Sim that only uses black-box access to the machine $\mathcal{M}$ encapsulated in the token.*

Proof (Sketch) For completeness, the (single-session) OT functionality $\mathcal{F}_{\mathsf{OT}}$ is given in Figure 2. Let π be a protocol between a sender S and a receiver R, in which a single token is sent from S to R. (The other case is handled analogously.) Consider the following environment $\mathcal{Z}'$ and the dummy adversary $\mathcal{A}'$ corrupting the sender: $\mathcal{Z}'$ chooses random bits s_0, s_1, and b, and instructs the sender to run π on input (s_0, s_1), including making the $\mathcal{F}_{\mathsf{wrap}}$ query to create a token encapsulating some program $\mathcal{M}$. Once R outputs $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, s)$, then $\mathcal{Z}'$ outputs 1 if $s = s_b$ and 0 otherwise.

Suppose that π securely realizes $\mathcal{F}_{\mathsf{OT}}$, where the proof of security uses a black-box simulator Sim (we stress that "black-box" here refers to the simulator's use of $\mathcal{M}$, not to its use of $\mathcal{Z}'$ or $\mathcal{A}'$ [which is always black box by definition of the UC framework]). Sim plays the role of a receiver while interacting with $\mathcal{A}'$, and is additionally given black-box access to whatever code $\mathcal{M}$ is encapsulated in the token by $\mathcal{A}'$. Note that at some point during its execution, Sim must send some inputs $(\tilde{s}_0, \tilde{s}_1)$ to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$. It is not hard to see that we must have $(\tilde{s}_0, \tilde{s}_1) = (s_0, s_1)$ with all but negligible probability.

We now consider a different environment $\mathcal{Z}$ and an adversary $\mathcal{A}$ corrupting the receiver. $\mathcal{Z}$ chooses random bits s_0, s_1, b and provides (s_0, s_1) as input to the (honest) sender; it outputs 1 iff $\mathcal{A}$ outputs (s_0, s_1). Adversary $\mathcal{A}$ works as follows:

> Run Sim, relaying messages from the honest sender to this internal copy of Sim. Whenever Sim wants to run $\mathcal{M}$ (the code embedded by the honest sender in the token that it creates) on some input q, adversary $\mathcal{A}$ simply submits q to the token that it received from the sender. (Formally, $\mathcal{A}$ sends $(\text{RUN}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, q)$ to $\mathcal{F}_{\mathsf{wrap}}$ and gives the response to Sim.) At some point, Sim sends $(\text{SEND}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \langle \tilde{s}_0, \tilde{s}_1\rangle)$ to $\mathcal{F}_{\mathsf{OT}}$, at which point $\mathcal{A}$ outputs $(\tilde{s}_0, \tilde{s}_1)$ and halts.

The key point is that *because the token is stateless*, there is no difference between Sim's black-box access to the code $\mathcal{M}$ (encapsulated in the token) and $\mathcal{A}$'s access to the token (via $\mathcal{F}_{\mathsf{wrap}}$). Thus, $\mathcal{A}$ provides a perfect simulation for Sim, and we conclude that $(\tilde{s}_0, \tilde{s}_1) = (s_0, s_1)$ with all but negligible probability in an execution of π in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. But this occurs with probability at most 1/2 in an ideal-world evaluation of $\mathcal{F}_{\mathsf{OT}}$. Thus, $\mathcal{Z}$ can distinguish between the real- and ideal-world executions, contradicting the claimed security of π. ■

6 Coin Tossing Using One Stateless Token

In the previous section we showed that universally composable OT cannot be realized from one stateless token if only "black-box techniques" are used. We complement that result by showing that UC secure computation from one stateless token *is* feasible via *non*black-box techniques. Here, we find it somewhat simpler to construct a protocol for universally composable coin tossing rather than OT. Note that coin tossing suffices for general secure computation under a variety of cryptographic assumptions [CLOS02, LPV09].

At a high level, our protocol follows the general structure of Blum's coin-tossing protocol [Blu82]. This protocol consists of three moves. In the first move (B1), Alice commits to a random x and sends com_x to Bob, who in return chooses a random value y and sends it to Alice (B2). In the last move (B3), Alice sends the decommitment to x and both parties output $x \oplus y$.

To obtain a UC coin-tossing protocol that follows this basic approach, we need to be able to *simulate* each party. In particular, if Alice is malicious then the simulator needs to extract the message contained in com_x (extractability), whereas when Bob is corrupted the simulator needs to open the commitment in an arbitrary way (equivocation). We achieve both goals by having Bob send a single stateless token $\mathcal{T}_{\mathsf{B}}$ to Alice. This token behaves in two different ways, depending on its input. The first task of the token is to generate a random value e upon seeing com_x (we discuss the details later); the second task is to generate a notification t for Bob that Alice knows the decommitment x. The value e is used for equivocation, and the notification t gives the simulator the ability to extract x. We give further details next. In the high-level description here, we assume the token is created honestly; we deal with a potentially malicious token when we formally define the protocol, below.

Achieving extractability. Similar to Section 4, the token only works on authenticated inputs. That is, whenever Alice wants to query the token on some inputs, she needs to first ask Bob to compute a MAC on those inputs. In order to achieve extractability, we let Alice query the token on input $(\mathsf{com}_x, x, r_x, \tau_x)$, where $\mathsf{com}_x \leftarrow \mathsf{Scom}(x; r_x)$ comes from Alice and τ_x is a MAC tag on com_x. The output of the token is a random value t that can be seen as a notification to Bob that Alice knows the decommitment x. As in the previous OT protocol, the authentication of the inputs guarantees that Alice makes exactly one valid query to the token: more than one valid query would imply that Alice has violated security of the MAC or binding of com_x; if Alice makes no query, then she cannot guess t. By forcing Alice to query the token exactly once, we can now easily construct a simulator that extracts the value x while emulating $\mathcal{F}_{\mathsf{wrap}}$. Modifying Blum's coin-tossing protocol, we have the following step:

(B1) Alice commits to x by executing $\mathsf{com}_x \leftarrow \mathsf{Scom}(x; r_x)$. She sends the commitment com_x to Bob, who in turn computes a tag τ_x on com_x and sends τ_x to Alice. Alice runs the token with $(\mathsf{com}_x, x, r_x, \tau_x)$ to obtain output t, and sends t to Bob. Bob checks if t is correct.

Achieving equivocation. Toward this goal, we further modify the protocol and the token. Alice sends com_x and a dummy commitment com_M before getting the tag τ_x on $\mathsf{com}_x \| \mathsf{com}_M$ from Bob. The token also gets as an additional input this commitment com_M; it outputs a random e on input $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ and a random t on input $(\mathsf{com}_x, \mathsf{com}_M, x, r_x, \tau_x)$. In step (B3), instead of sending the decommitment (x, r_x) of com_x to Bob, Alice sends x together with a witness-indistinguishable (WI) proof that either x is a valid decommitment of com_x, or com_M contains

On input $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ do:
  if $\mathsf{Vrfy}_s(\mathsf{com}_x \| \mathsf{com}_M, \tau_x) = 1$
    output e
  else output $\perp$

On input $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ do:
  if $\mathsf{Vrfy}_s(\mathsf{com}_x \| \mathsf{com}_M, \tau_x) = 1$ and $x = \mathsf{Open}(\mathsf{com}_x, x, r_x)$
    output (t, r_t)
  else output $(\perp, \perp)$

Figure 8: The Turing machine $\mathcal{M}$ embedded in the sender-created token $\mathcal{T}_{\mathsf{B}}$.

code that outputs the actual output e of the token $\mathcal{T}_{\mathsf{B}}$. (This is where we use the nonblack-box techniques of Barak [Bar01].)
Due to the binding property of com_M, and because the notification e is unpredictable, Alice cannot commit to such code in com_M. The simulator, however, takes advantage of the fact that it obtains the code of the token generated by Bob while emulating $\mathcal{F}_{\mathsf{wrap}}$. Then, as in [Bar01], the simulator's ability to predict the output of the token beforehand can be used to achieve equivocation. We remark that in contrast to Barak's work, we do not need to use universal arguments; this is because $\mathcal{F}_{\mathsf{wrap}}$ is parameterized with a fixed polynomial bounding the running time of the token (whereas Barak had to handle any polynomial running time).

More formally, we change the protocol as follows:

Token The token input is either $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ or $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and in both cases it checks the validity of τ_x before responding. In the first case, the token outputs a random value e. In the second case, it returns a random value t.

Protocol Once Alice obtains the token, she runs the following protocol with Bob:

(B1) Alice commits to x and sends com_x together with a dummy commitment com_M to Bob. In turn, Bob authenticates $\mathsf{com}_x \| \mathsf{com}_M$ using a MAC and sends (τ_x, e) to Alice. Alice invokes the token twice, first with $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ and then with $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$, obtaining the values e' and t', respectively. She checks if $e = e'$ and, if so, sends t' to Bob. Finally, Bob checks if $t' = t$.

(B2) Bob sends Alice a random value y.

(B3) Alice sends x together with a WI proof that either (i) (x, r_x) is the decommitment of com_x, or (ii) com_M is a commitment to a Turing machine $\mathcal{M}$ such that $\mathcal{M}(\mathsf{com}_x, \mathsf{com}_M, \tau) = e$.

6.1 Formal Description of the Protocol

The protocol ψ between Alice A and Bob B consists of an initial token-exchange phase, followed by a coin-tossing phase for generating a random λ-bit string. We now describe ψ formally; see also Figure 9.

Token-exchange phase. Bob generates a single token $\mathcal{T}_{\mathsf{B}}$ and sends it to Alice. Bob's token $\mathcal{T}_{\mathsf{B}}$ encapsulates the code $\mathcal{M}$ described in Figure 8, where s, e, t, r_t are each chosen uniformly from $\{0,1\}^\lambda$.

Coin-tossing phase. In this phase, Alice and Bob proceed as follows.

$\mathsf{B} \to \mathsf{A}$**:** Bob chooses a collision-resistant hash function $H \leftarrow \mathcal{H}$. He also commits to the value t (that he used when creating the token) by executing $\mathsf{com}_t \leftarrow \mathsf{Com}(t; r_t)$. He sends H, com_t to Alice.

$\mathsf{A} \to \mathsf{B}$: Alice chooses a value $x \leftarrow \{0,1\}^{2\lambda}$ and commits to x and $H(0^\lambda)$ by executing $\mathsf{com}_x \leftarrow \mathsf{Scom}(x; r_x)$ and $\mathsf{com}_M \leftarrow \mathsf{Com}(H(0^\lambda); r_M)$. She sends com_x and com_M to Bob.

$\mathsf{B} \to \mathsf{A}$: Bob generates a MAC tag $\tau_x := \mathsf{Tag}_s(\mathsf{com}_x \| \mathsf{com}_M)$, and sends (e, τ_x) to Alice.

$\mathsf{A} \to \mathsf{B}$: Alice runs the token $\mathcal{T}_\mathsf{B}$ with $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ and obtains e' in response. Then she runs the token with $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and obtains (t', r'_t). Alice checks if $e' = e$ and $t' = \mathsf{Open}(\mathsf{com}_t, t', r'_t)$. If not, she aborts the protocol. Otherwise, she chooses $v \leftarrow \mathsf{Seed}$ and sends (t', v) to Bob.

$\mathsf{B} \to \mathsf{A}$: Bob checks that $t = t'$, and aborts if not. Otherwise he chooses $y \leftarrow \{0,1\}^\lambda$ and sends it to Alice.

$\mathsf{A} \to \mathsf{B}$: Alice sends $\tilde{x} := \mathsf{Ext}(x, v)$ and gives a WI argument of knowledge that $(H, \mathsf{com}_x, \mathsf{com}_M, e, \tau_x, v, \tilde{x})$ belongs to the $\mathcal{NP}$ language L defined by the following relation R_L:

$R_L((H, \mathsf{com}_x, \mathsf{com}_M, e, \tau_x, v, \tilde{x}), (\alpha, \beta)) = 1$ if either one of the following holds:

(i) $\alpha = \mathsf{Open}(\mathsf{com}_x, \alpha, \beta)$ and $\tilde{x} = \mathsf{Ext}(\alpha, v)$.

(ii) $H(\alpha) = \mathsf{Open}(\mathsf{com}_M, H(\alpha), \beta)$. In addition, treating α as the description of a Turing machine, the execution of $\alpha(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ outputs e in time at most $p(\lambda)$, where p is the polynomially bounded running time defined by $\mathcal{F}_\mathsf{wrap}$.

If the proof succeeds, both parties output $\tilde{x} \oplus y$.

Theorem 4 *Assume* Com *is computationally hiding and strongly binding,* Scom *is statistically hiding and computationally binding,* MAC *is a deterministic, unforgeable message-authentication code, $\mathcal{H}$ is a family of collision-resistant hash functions, and the proof system is a witness-indistinguishable argument of knowledge. Then ψ securely realizes $\mathcal{F}_\mathsf{coin}$ in the $\mathcal{F}_\mathsf{wrap}$-hybrid model.*

To show the security of the protocol, we need to construct a simulator Sim for any PPT environment $\mathcal{Z}$ such that $\text{EXEC}^{\mathcal{F}_\mathsf{wrap}}_{\mathcal{A},\pi,\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_\mathsf{coin},\mathsf{Sim},\mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary. Below, we briefly provide ideas for simulation.

Corrupting Alice. Note that Alice cannot commit to a TM $\mathcal{M}$ in com_M that will output e, since e is chosen at random independently. Therefore, from the collision-resistance of H, Alice has to show that $\tilde{x} = \mathsf{Ext}(x, v)$ in the WI proof. Since v appears in the communication transcript, by forcing malicious Alice A^* to query $\mathcal{T}_\mathsf{B}$ with x exactly once, the simulator Sim can extract the value $\tilde{x}$ from binding of com_x. Then, upon receiving the random string coins from $\mathcal{F}_\mathsf{coin}$, the simulator can send $y = \tilde{x} \oplus \mathsf{coins}$ to A^*. To see how the protocol forces exactly one query to the token, note that malicious Alice is not able to generate a valid t without querying $\mathcal{T}_\mathsf{B}$, since t is random and com_t is hiding. Also, note that A^* cannot query $\mathcal{T}_\mathsf{B}$ with different xs without contradicting the unforgeability of the tag τ_x and/or the binding of com_x. This allows the simulator to extract the value of x.

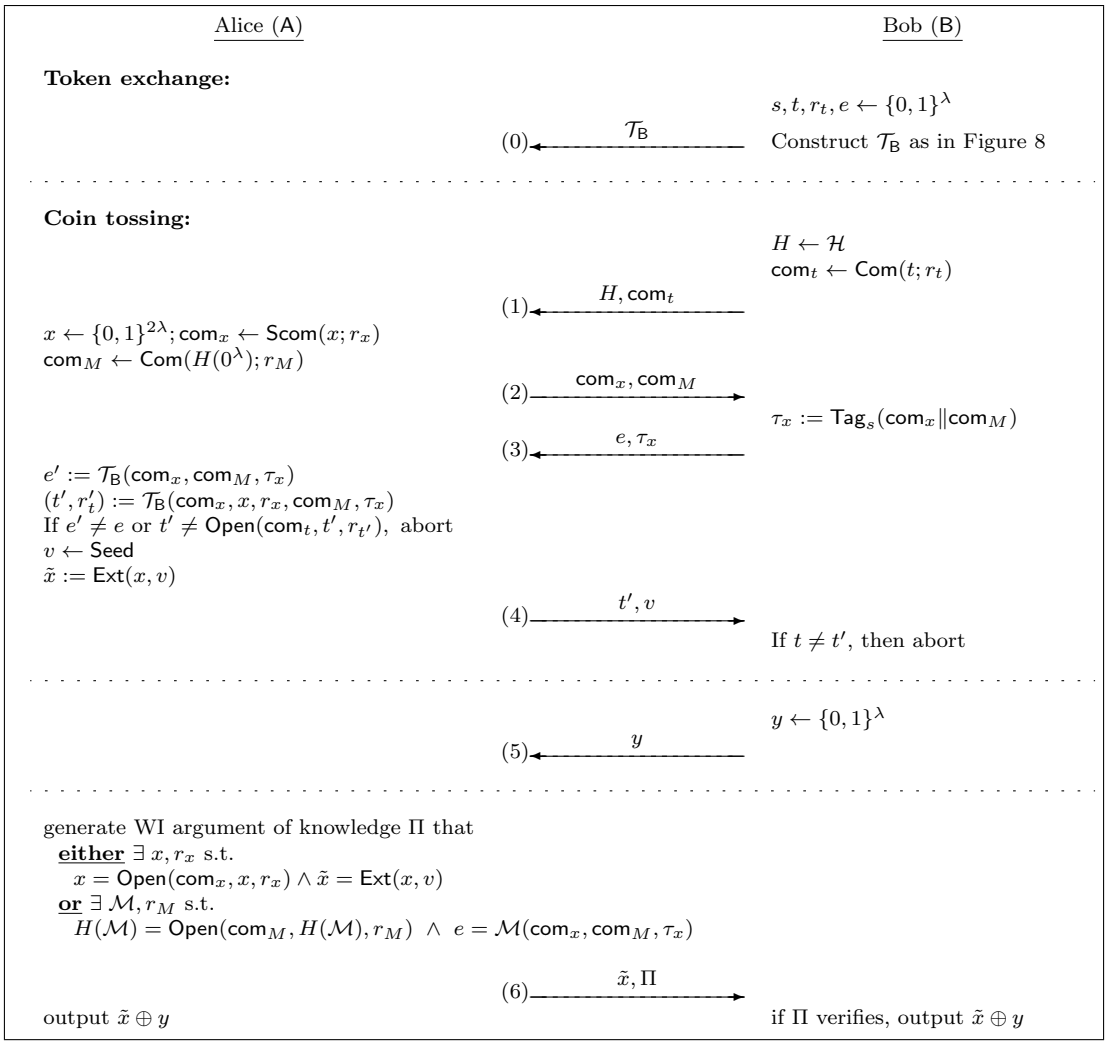

Figure 9: A coin-tossing protocol ψ from a single stateless token.

Corrupting Bob. The simulator Sim needs to equivocate the value $\tilde{x}$ so that it may hold that $\tilde{x} = \mathsf{coins} \oplus y$, where coins is the random string from $\mathcal{F}_{\mathsf{coin}}$. This is achieved as follows. While emulating $\mathcal{F}_{\mathsf{wrap}}$, the simulator obtains the token code $\mathcal{M}$ generated by Bob, then it generates $\mathsf{com}_M \leftarrow \mathsf{Com}(H(\mathcal{M}); R_M)$. Given the hiding property of com_M, the simulated transcript is indistinguishable from that in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. Now, with the witness $(\mathcal{M}, R_M)$ in the WI proof, the simulator can send any value $\tilde{x}$.

6.2 Corrupted Alice

Let $\mathcal{A}$ be the dummy adversary for Alice. We present the ideal-world simulator Sim such that for any environment $\mathcal{Z}$ it holds that $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\mathcal{A},\pi,\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\mathsf{coin}},\mathsf{Sim},\mathcal{Z}}$. The simulator Sim internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. The simulator Sim plays the role of Bob to interact with $\mathcal{A}$ as in the protocol except the following.

Initial Token Exchange:

1. Simulating $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ as specified in the protocol, return $(\text{READY}, \langle \mathsf{sid}, \mathsf{B}, \mathsf{A} \rangle)$ to $\mathcal{A}$.[5]

Coin Tossing:

1. Simulate $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ as specified in the protocol with the following exception: When $\mathcal{A}$ makes a valid query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$, then record x; however, if $\mathcal{A}$ makes two different valid queries $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and $(\mathsf{com}'_x, x', r'_x, \mathsf{com}'_M, \tau'_x)$, then abort.
2. Upon receiving (t', v) in step (4) from $\mathcal{A}$, if $t = t'$, then compute $\tilde{x} := \mathsf{Ext}(x, v)$ based on previously recorded x (if there is no x recorded, then abort). In addition, in the name of corrupted Alice, send $(\text{TOSS}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle)$ to functionality $\mathcal{F}_{\mathsf{coin}}$ and receive $(\text{COIN}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins})$ from the functionality. Then compute $y := \mathsf{coins} \oplus \tilde{x}$.

We show computational indistinguishability between the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and the ideal world by the following hybrids.

Hybrid $\mathcal{H}_0$: This is the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world execution $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\mathcal{A},\pi,\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: This is $\text{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_1,\mathcal{Z}}$. The functionality $\mathcal{F}_1$ is the same as $\mathcal{F}_{\mathsf{coin}}$ except that output string coins is not chosen by itself, rather it is from the simulator. That is, upon receiving $(\text{GO}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{B}, \mathsf{coins})$ from the simulator, $\mathcal{F}_1$ stores coins, and later upon receiving $(\text{TOSS}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle)$ from Bob, $\mathcal{F}_1$ sends $(\text{COIN}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins})$ to Bob. The simulator Sim_1 works as follows:

- The simulator Sim_1 internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. In addition, Sim simulates the following interactions with the internal $\mathcal{A}$.
- The simulator Sim_1 plays as Bob following the protocol honestly while it is interacting with $\mathcal{A}$ playing as Alice. In addition, when the execution has ended, the simulator sends its output $\mathsf{coins} := \tilde{x} \oplus y$ to $\mathcal{F}_1$ by sending $(\text{GO}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{B}, \mathsf{coins})$.

[5] We sometimes let $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ denote the $\mathcal{F}_{\mathsf{wrap}}$ functionality that manages the token $\mathcal{T}_{\mathsf{B}}$.

- The simulator Sim_1 simulates $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$ honestly, only observing the queries of $\mathcal{A}$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$.

ANALYSIS: Obviously, $\mathcal{H}_0$ and $\mathcal{H}_1$ are identically distributed.

Hybrid $\mathcal{H}_2$: This is $\text{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_2,\mathcal{Z}}$. The simulator Sim_2 works the same as Sim_1, except that Sim_2 outputs abort if $\mathcal{A}$ sends the correct t' while making zero valid query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$.

ANALYSIS: We show that the probability that Sim_2 outputs abort is negligible in λ. Roughly speaking, from the hiding property of com_t, this probability is negligible. In particular, given the dummy adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ that makes Sim_2 output abort with non-negligible probability, one can easily construct a non-uniform adversary $\mathcal{B}$ that breaks the hiding property of the commitment scheme as follows:

> $\mathcal{B}$ chooses two messages (m_0, m_1) to break the hiding property, where m_0 is 0^λ and m_1 is t. Remember that this t is chosen uniformly at random from $\{0,1\}^\lambda$. Given the commitment com to m_0 or m_1, $\mathcal{B}$ works as Sim_2 using com as com_t. If Sim_2 aborts because of $\mathcal{A}$ making no valid query (but outputting the correct t), then $\mathcal{B}$ outputs 1; otherwise $\mathcal{B}$ outputs 0.

If com is a commitment to m_0, the probability that $\mathcal{B}$ outputs 1 is exactly $2^{-\lambda}$, since t is chosen randomly and independently of the view of $\mathcal{A}$ and $\mathcal{Z}$. On the other hand, if com is a commitment to m_1, $\mathcal{B}$ outputs 1 with non-negligible probability from the assumption about $\mathcal{A}$ and $\mathcal{Z}$. Hence, $\mathcal{B}$ has a non-negligible advantage in breaking the hiding property of the commitment.

Hybrid $\mathcal{H}_3$: This is $\text{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_3,\mathcal{Z}}$. The simulator Sim_3 works the same as Sim_2 except that Sim_3 outputs abort if $\mathcal{A}$ makes two different valid queries $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and $(\mathsf{com}'_x, x', r'_x, \mathsf{com}'_M, \tau'_x)$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$.

ANALYSIS: We show that the probability that Sim_3 outputs abort is negligible in λ.

Note that this event occurs when $(\mathsf{com}_x, \mathsf{com}_M, \tau_x) \neq (\mathsf{com}'_x, \mathsf{com}'_M, \tau'_x)$ or when $(x, r_x) \neq (x', r'_x)$. The first case directly violates the unforgeability of the deterministic canonical MAC, and the second case the binding property of the commitment scheme. Thus, both cases happen with negligible probability.

Hybrid $\mathcal{H}_4$: This is $\text{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_4,\mathcal{Z}}$. The simulator Sim_4 works the same as Sim_3 except the following:

- The simulator Sim_4 records the value x in the valid query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ that $\mathcal{A}$ makes to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$,
- The simulator Sim_4 outputs abort if the proof Π is accepted but $\tilde{x} \neq \mathsf{Ext}(x, v)$.

ANALYSIS: We show that the probability that Sim_4 outputs abort is negligible in λ.

We first argue that, with all but negligible probability, the environment $\mathcal{Z}$ and the dummy adversary $\mathcal{A}$ are not able to generate an accepting proof using $\mathcal{M}, r_M$ as a witness. That is, they can not give a argument of knowledge of $\mathcal{M}, r_M$ such that $H(\mathcal{M}) = \mathsf{Open}(\mathsf{com}_M, H(\mathcal{M}), r_M)$ and $e = \mathcal{M}(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$. If this is not the case, we can construct a non-uniform adversary

$\mathcal{B}$ that breaks the collision-resistance of the given hash family $\mathcal{H}$. In particular, let $\mathcal{Z}^*$ be the environment that uses the witness $\mathcal{M}, r_M$ with non-negligible probability. Without loss of generality, we can assume that $\mathcal{Z}^*$ is deterministic (with randomness encoded in the advice). $\mathcal{B}$ is given a hash function H from outside. $\mathcal{B}$ works as follows:

> $\mathcal{B}$ works as Sim_4 with $\mathcal{Z}^*$ while using the hash function H given from outside. When the proof Π succeeds, it extracts the witness that $\mathcal{Z}^*$ used, by using the extractor of the WI argument of knowledge. Then, $\mathcal{B}$ rewinds the above execution right after $\mathcal{Z}^*$ has sent $(\mathsf{com}_x, \mathsf{com}_M)$, and then it continues the execution as before only except that this time it uses another random value e_2 in the protocol and the simulation of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$. When the proof Π succeeds, it extracts the witness that $\mathcal{Z}^*$ used.

Let δ be the probability that $\mathcal{Z}^*$ succeeds, making Sim_4 abort. Then, for at least a $\delta/2$ fraction of the $H \in \mathcal{H}$, it holds that $\mathcal{A}$ is able to output a witness for the transcript with this function H with probability $\delta/2$. Fix such an $H \in \mathcal{H}$. Now, since we choose e randomly, it holds that with probability $\delta/2$, $\mathcal{B}$ can extract the code $\mathcal{M}$ such that com_M is a commitment to $H(\mathcal{M})$ and $\mathcal{M}(\mathsf{com}_x, \mathsf{com}_M, \tau_x) = e$. Moreover, since we also choose e_2 randomly, it holds that with probability $\delta/2$, $\mathcal{B}$ will also extract the code $\mathcal{M}_2$ such that com_M is a commitment to $H(\mathcal{M}_2)$ such that $\mathcal{M}_2(\mathsf{com}_x, \mathsf{com}_M, \tau_x) = e_2$. Note that the same com_M is used, and $e \neq e_2$ implies $\mathcal{M} \neq \mathcal{M}_2$. Therefore, $\mathcal{B}$ can get a collision $(\mathcal{M}, \mathcal{M}_2)$ such that $H(\mathcal{M}) = H(\mathcal{M}_2)$ with non-negligible probability $\delta^3/8 - \mathsf{negl}(\lambda)$.

It is left to show that it occurs with negligible probability that $\mathcal{Z}$ uses the witness (x', r'_x) such that $x' = \mathsf{Open}(\mathsf{com}_x, x', r'_x)$ and $\tilde{x} = \mathsf{Ext}(x', v)$ in the proof Π. Note that $\tilde{x} \neq \mathsf{Ext}(x, v)$ implies $x \neq x'$. Therefore, if this event occurs with non-negligible probability, we can construct an algorithm that breaks the binding of com_x by extracting the witness used in the proof Π. Therefore, we conclude that Sim_4 outputs abort with negligible probability.

Hybrid $\mathcal{H}_5$: This is $\text{IDEAL}_{\mathcal{F}_{\mathsf{coin}}, \mathsf{Sim}_5, \mathcal{Z}}$. The simulator Sim_5 is the same as Sim_4 except the following:

- Sim_5 is interacting with $\mathcal{F}_{\mathsf{coin}}$ instead of interacting with $\mathcal{F}_1$. Now, Sim_5 receives a random string coins from $\mathcal{F}_{\mathsf{coin}}$. Then, Sim_5 computes $y := \mathsf{coins} \oplus \mathsf{Ext}(x, v)$.

We note that Sim_5 is the simulator Sim described before.

ANALYSIS: Note that y in $\mathcal{H}_5$ is still uniformly random since coins obtained from $\mathcal{F}_{\mathsf{coin}}$ is uniformly random. Therefore, $\mathcal{H}_4$ and $\mathcal{H}_5$ are identically distributed.

6.3 Corrupted Bob

Let $\mathcal{A}$ be the dummy adversary for Bob. We present the ideal-world simulator Sim such that for any environment $\mathcal{Z}$ it holds that $\text{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\mathsf{coin}}, \mathsf{Sim}, \mathcal{Z}}$. The simulator Sim internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. The simulator Sim plays the role of Alice to interact with $\mathcal{A}$ as in the protocol except the following.

Initial Token Exchange:

1. Simulate a copy of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$. Upon receiving $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{B}, \mathsf{A} \rangle, \mathcal{M})$ from $\mathcal{A}$ to the simulated $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$, store $\mathcal{M}$.

Coin Tossing:

1. Simulate $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$ by directly running $\mathcal{M}$.
2. In the real protocol, com_M sent in step (2) is a commitment to 0^λ, i.e., $\mathsf{com}_M \leftarrow \mathsf{Com}(H(0^\lambda); r_M)$. Here, com_M is instead a commitment to the code $\mathcal{M}$ (if the code is stateful, Sim hardwares the state into $\mathcal{M}$), i.e., compute $\mathsf{com}_M \leftarrow \mathsf{Com}(H(\mathcal{M}); r_M)$.
3. Upon receiving y in step (5) from $\mathcal{A}$, send in the name of corrupted Bob (TOSS, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle$) to $\mathcal{F}_{\mathsf{coin}}$, and receive (COIN, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins}$) from $\mathcal{F}_{\mathsf{coin}}$. Compute $\tilde{x} := y \oplus \mathsf{coins}$ and send $(\tilde{x}, \Pi)$ to $\mathcal{A}$, where the proof Π is generated by using the witness $(\mathcal{M}, r_M)$ instead of (x, r_x).

We show computational indistinguishability between the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and the ideal world by the following hybrids.

Hybrid $\mathcal{H}_0$: This is the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world execution $\mathrm{EXEC}^{\mathcal{F}_{\mathsf{wrap}}}_{\pi,\mathcal{A},\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: This is $\mathrm{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_1,\mathcal{Z}}$. Here $\mathcal{F}_1$ is the same as $\mathcal{F}_{\mathsf{coin}}$ except that the output string coins is not chosen by itself, but rather it is from the simulator. That is, upon receiving (GO, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{A}, \mathsf{coins}$) from the simulator, $\mathcal{F}_1$ stores coins, and later upon receiving (TOSS, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle$) from the dummy Alice, $\mathcal{F}_1$ returns (COIN, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins}$) to her. The simulator Sim_1 works as follows:

- The simulator Sim_1 internally simulates a copy of the dummy adversary $\mathcal{A}$ as well as $\mathcal{A}$'s communication with the external environment $\mathcal{Z}$, that is each message from/to the environment $\mathcal{Z}$ is forwarded to/from the internal $\mathcal{A}$. In addition, Sim simulates the following interactions with the internal $\mathcal{A}$ by honestly following the protocol specification of Alice.
- During this internal execution, Sim_1 simulates $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$ honestly. In particular, upon receiving (CREATE, $\langle \mathsf{sid}, \mathsf{B}, \mathsf{A} \rangle, \mathcal{M}$) from $\mathcal{A}$ to the simulated $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$, store $\mathcal{M}$. Thereafter, $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{B}}$ is simulated by directly running $\mathcal{M}$.
- When the execution with $\mathcal{A}$ has ended, the simulator sends its output $\mathsf{coins} := \tilde{x} \oplus y$ to $\mathcal{F}_1$ by sending (GO, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{A}, \mathsf{coins}$).

ANALYSIS: Obviously, $\mathcal{H}_0$ and $\mathcal{H}_1$ are identically distributed.

Hybrid $\mathcal{H}_2$: This is $\mathrm{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_2,\mathcal{Z}}$. Here, the simulator Sim_2 is the same as Sim_1 except that com_M in step (2) is not based on 0^λ, but on the code $\mathcal{M}$ (if the code is stateful, Sim_1 hardwires the state into $\mathcal{M}$). I.e., Sim_2 computes $\mathsf{com}_M \leftarrow \mathsf{Com}(H(\mathcal{M}); r_M)$.

ANALYSIS: Roughly speaking, from the hiding property of com_M, the two distributions are indistinguishable. Suppose there are the dummy adversary $\mathcal{A}$ and an environment $\mathcal{Z}^*$ that distinguishes the two distributions with non-negligible probability. We show a non-uniform polynomial time algorithm $\mathcal{B}$ that breaks the computational hiding property of the commitment scheme.

> $\mathcal{B}$ chooses two messages (m_0, m_1) to break the hiding property: m_0 is $H(0^\lambda)$, and m_1 is $H(\mathcal{M})$ from $\mathcal{A}$. Given the commitment com to m_0 or m_1, $\mathcal{B}$ runs a protocol

with $\mathcal{A}$ and $\mathcal{Z}^*$ while working as Sim_1 (or Sim_2) using com as com_M. This is possible since $\mathcal{B}$ does not use the decommitment of com_M but the decommitment (x, r_x) of com_x as a witness in the proof Π. Finally, $\mathcal{B}$ outputs whatever $\mathcal{Z}^*$ outputs.

If com is a commitment to m_0, then $\mathcal{A}$ and $\mathcal{Z}^*$ run exactly in the hybrid $\mathcal{H}_1$. On the other hand, if com is commitment to m_1, $\mathcal{A}$ and $\mathcal{Z}^*$ run exactly in the hybrid $\mathcal{H}_2$. Therefore, $\mathcal{B}$ can break the hiding with non-negligible probability.

Hybrid $\mathcal{H}_3$: This is $\text{IDEAL}_{\mathcal{F}_1,\mathsf{Sim}_3,\mathcal{Z}}$. Here, the simulator Sim_3 is same as Sim_2 except that the proof Π is generated by using the witness $(\mathcal{M}, r_M)$.

ANALYSIS: $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable due to the WI property.

Hybrid $\mathcal{H}_4$: This is $\text{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}_4,\mathcal{Z}}$. The simulator Sim_4 is same as Sim_3 except the following.

- Upon receiving y in step (5) from $\mathcal{A}$, send in the name of corrupted Bob (TOSS, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle$) to $\mathcal{F}_{\mathsf{coin}}$, and receive (COIN, $\langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{coins}$) from $\mathcal{F}_{\mathsf{coin}}$. Compute $\tilde{x} := y \oplus \mathsf{coins}$ and send $(\tilde{x}, \Pi)$ to $\mathcal{A}$, where the proof Π is generated by using the witness $(\mathcal{M}, r_M)$.

We note that Sim_4 is the simulator Sim described before.

ANALYSIS: We show that $\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically close. In particular, it suffices to show that the distribution of $\tilde{x}$ in $\mathcal{H}_3$ is statistically close to that in $\mathcal{H}_4$. That is, we need to show that the distribution of $\mathsf{Ext}(x, v)$ is statistically close to the uniform distribution, conditioned on the distribution of the other variables in the transcript.

We consider possible information leakage from the maliciously generated token. Recall that it takes two inputs: $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ and $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$. The first input was already observed by $\mathcal{A}$, and thus the token output on it doesn't convey any extra information to $\mathcal{A}$. For the second input, due to the strong binding property of com_t, and because com_t (and thereby t from the strong binding) was chosen before x in the protocol, the malicious token may reveal at most one bit information on x only through aborting or not. Together with the statistical hiding property of com_x, this implies that the variable x has min-entropy at least $2\lambda - 1$ to the view of $\mathcal{A}$ and $\mathcal{Z}$. Therefore, with the strong extractor Ext, the distribution of $\mathsf{Ext}(x, v)$ is statistically close to the uniform distribution.

References

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *4th Theory of Cryptography Conference — TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, 2007.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 106–115. IEEE, 2001.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 186–195. IEEE, 2004.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In *Advances in Cryptology — Crypto 2011*, volume 6841 of *LNCS*, pages 51–70. Springer, 2011.

[Blu82] Manuel Blum. Coin flipping by telephone. In *Proc. IEEE COMPCOM*, pages 133–137, 1982.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Advances in Cryptology — Crypto '93*, volume 773 of *LNCS*, pages 302–318. Springer, 1994.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001. Latest version available at `http://eprint.iacr.org/2000/067`, December 2005.

[Can07] Ran Canetti. Obtaining universally compoable security: Towards the bare bones of trust (invited talk). In *Advances in Cryptology — Asiacrypt 2007*, volume 4833 of *LNCS*, pages 88–112. Springer, 2007.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, 2001.

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 545–562. Springer, 2008.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *Journal of Cryptology*, 19(2):135–167, 2006.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, 2002.

[CP93a] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — Crypto '92*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[CP93b] Ronald Cramer and Torben P. Pedersen. Improved privacy in wallets with observers (extended abstract). In *Advances in Cryptology — Eurocrypt '93*, volume 765 of *LNCS*, pages 329–343. Springer, 1993.

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *48th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 249–259. IEEE, 2007.

[DKM11] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *8th Theory of Cryptography Conference — TCC 2011*, volume 6597 of *LNCS*, pages 164–181. Springer, 2011.

[DMMQN11] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Basing obfuscation on simple tamper-proof hardware assumptions. Cryptology ePrint Archive, Report 2011/675, 2011. `http://eprint.iacr.org/`.

[DNW09] Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Universally composable multiparty computation with partially isolated parties. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 315–331. Springer, 2009.

[DPP97] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, 1997.

[DQ87] Yvo Desmedt and Jean-Jacques Quisquater. Public-key systems based on the difficulty of tampering (is there a difference between DES and RSA?). In *Advances in Cryptology — Crypto '86*, volume 263 of *LNCS*, pages 111–117. Springer, 1987.

[DSV10] Maria Dubovitskaya, Alessandra Scafuro, and Ivan Visconti. On efficient non-interactive oblivious transfer with tamper-proof hardware. Cryptology ePrint Archive, Report 2010/509, 2010.

[FPS+11] Marc Fischlin, Benny Pinkas, Ahmad-Reza Sadeghi, Thomas Schneider, and Ivan Visconti. Secure set intersection with untrusted hardware tokens. In *Cryptographers' Track — RSA 2011*, volume 6558 of *LNCS*, pages 1–16. Springer, 2011.

[FS90] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *Advances in Cryptology — Crypto '89*, volume 435 of *LNCS*, pages 526–544. Springer, 1990.

[GGM85] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Advances in Cryptology — Crypto '84*, volume 196 of *LNCS*, pages 276–288. Springer, 1985.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Advances in Cryptology — Crypto 2010*, volume 6223 of *LNCS*, pages 173–190. Springer, 2010.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, 2010.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HK05] Omer Horvitz and Jonathan Katz. Bounds on the efficiency of "black-box" commitment schemes. In *ICALP 2005: 32nd Intl. Colloquium on Automata, Languages, and Programming (ICALP)*, volume 3580 of *LNCS*, pages 128–139. Springer, 2005.

[HL08] C. Hazay and Y. Lindell. Constructions of truly practical secure protocols using standard smartcards. In *ACM CCS '08: 15th ACM Conf. on Computer and Communications Security*, pages 491–500. ACM Press, 2008.

[HMU05] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *Proc. 5th Central European Conference on Cryptology (MoraviaCrypt)*, 2005.

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM Press, 2007.

[JKSS10] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Embedded SFE: Offloading server and network using hardware tokens. In *Financial Cryptography and Data Security 2010*, volume 6052 of *LNCS*, pages 207–221. Springer, 2010.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *Advances in Cryptology — Eurocrypt 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, 2007.

[Kol10] Vladimir Kolesnikov. Truly efficient string oblivious transfer using resettable tamper-proof tokens. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 327–342. Springer, 2010.

[Lin09] Yehuda Lindell. General composition and universal composability in secure multi-party computation. *Journal of Cryptology*, 22(3):395–428, 2009.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: Universal composability from stand-alone non-malleability. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 179–188. ACM Press, 2009.

[MS08] Tal Moran and Gil Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 527–544. Springer, 2008.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.