(Efficient) Universally Composable Oblivious Transfer Using a Minimal Number of Stateless Tokens[*]

Seung Geol Choi[†] Jonathan Katz[‡] Dominique Schröder[§] Arkady Yerukhimovich[¶]
Hong-Sheng Zhou[||]

Abstract

We continue the line of work initiated by Katz (Eurocrypt 2007) on using *tamper-proof hardware tokens* for universally composable secure computation. As our main result, we show an oblivious-transfer (OT) protocol in which two parties each create and transfer a single, stateless token and can then run an unbounded number of OTs. We also show a more efficient protocol, based only on standard symmetric-key primitives (block ciphers and collision-resistant hash functions), that can be used if a *bounded* number of OTs suffice.

Motivated by this result, we investigate the number of stateless tokens needed for universally composable OT. We prove that our protocol is *optimal* in this regard for constructions making *black-box* use of the tokens (in a sense we define). We also show that nonblack-box techniques can be used to obtain a construction using only a single stateless token.

[*] Invited to the *Journal of Cryptology* by the program committee of TCC 2014.

[†] United States Naval Academy. Work done in part at the University of Maryland and Columbia University, and supported in part by ONR award N0001416WX01645 and NSF award #1618269. Email: `choi@usna.edu`.

[‡] University of Maryland. Work supported in part by NSF awards #1111599 and #1223623 as well as a Humboldt Research Fellowship. Email: `jkatz@cs.umd.edu`.

[§] Friedrich-Alexander University Erlangen-Nürnberg. Work done in part at the University of Maryland and Saarland University, and supported by Bavaria at the Nuremberg Campus of Technology (NCT), the BMBF project PROMISE (FKZ:16KIS0362K), the German Research Foundation (DFG) via the collaborative research center "Methods and Tools for Understanding and Controlling Privacy" (SFB 1223), the German Ministry for Education and Research (BMBF) through funding for the Center for IT-Security, Privacy, and Accountability (CISPA, www.cispa-security.org), and an Intel Early Career Award. Email: `dosc@cs.fau.de`.

[¶] MIT Lincoln Laboratory. Work done in part at the University of Maryland. Email: `arkady@cs.umd.edu`.

[||] Virginia Commonwealth University. Work done in part at the University of Maryland, and supported by an NSF Computing Innovation postdoctoral fellowship. Email: `hszhou@vcu.edu`.

1 Introduction

The universal composability (UC) framework [7] 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, most interesting cryptographic tasks are impossible to realize in the "plain" UC framework when an honest majority cannot be assumed and, in particular, in the setting of two-party computation [9, 10, 44]. This negative result has motivated researchers to explore various extensions/variants of the plain UC framework in which secure computation can be achieved [8], with notable examples being the assumption of a common reference string (CRS) [7, 9, 11] or a public-key infrastructure [7, 4]. 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 [12] for some ideas on using a naturally occurring high-entropy source in place of a CRS).

Katz [39] suggested using *tamper-proof hardware tokens* for UC computation. In this model, it is assumed that parties can construct hardware tokens to compute functions of their choice; an adversary given an honestly created token $\mathcal{T}_F$ for a function F can do no more than observe the input/output behavior of this token. The motivation for this model is that the existence of tamper-proof hardware can be viewed, in principle, as a *physical* assumption rather than an assumption of trust in some external entity. (In particular, parties can create the tamper-proof tokens themselves rather than obtain them from a trusted provider as in [34], and there is no assumption regarding what code a malicious party puts in a token it creates.) Secure hardware may also potentially result in more efficient protocols; indeed, it has been suggested for improving efficiency in other settings (e.g., [18, 14, 15, 6, 31, 38, 42, 23]). In addition to introducing the model, Katz showed that tamper-proof hardware tokens can be used for universally composable computation of arbitrary functions under additional cryptographic assumptions. His work motivated an extensive amount of follow-up work [13, 47, 26, 16, 28, 29, 20] that we discuss in detail later.

We show here two efficient protocols based on tamper-proof hardware tokens, and secure against a static, malicious adversary, for universally composable 1-out-of-2 string oblivious transfer (OT). They have the following advantages:

- Our protocols are based on *stateless* tokens, which seem easier/cheaper to create in practice and are (inherently) resistant to resetting attacks. Security holds even if maliciously generated tokens are stateful.
- Our protocols require the parties to exchange a *single* pair of tokens. This can be done in advance, before the parties' inputs are known. The number of tokens is optimal; see below.
- Our protocols are *black-box* and run in constant rounds.
- In our first protocol, the tokens can be used to implement an *unbounded* number of OTs, rather than requiring the parties to exchange a fresh pair of tokens for every oblivious transfer they wish to compute. Thus, by relying on known completeness results [41, 37], the parties can use the same tokens to perform an unlimited number of secure computations (of possibly different functions, and on different inputs).
- Our second protocol can be based on standard symmetric-key primitives (namely, block ciphers and collision-resistant hash functions) and does not use public-key techniques. However, it requires the parties to fix some upper bound on the number of OTs to be realized from

a single pair of exchanged tokens. (Though in this case *OT extension* [36] could be used to obtain an unbounded number of OTs, at the expense of a stronger cryptographic assumption.)

Inspired by the above, we investigate the minimal number of *stateless* tokens needed for universally composable OT/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 protocols are thus optimal in this regard. Note that a single *stateful* token suffices for universally composable OT [20]; our result thus demonstrates a separation 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 nonblack-box protocol for carrying out an unbounded number of secure computations (and hence OTs) using only a *single* stateless token. Our construction uses a variant of the simulation technique introduced by Barak [3].

1.1 Prior Work

Katz's original protocol for secure computation using tamper-proof tokens [39] 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. [13] were the first to eliminate the requirement of stateful tokens. They construct UC commitments assuming one-way functions, and oblivious transfer based on any enhanced trapdoor permutation (eTDP). They also introduce a stronger security model in which an adversary need not know the code of the tokens it produces, thus capturing scenarios where an adversary may pass along tokens whose code it does not know, e.g., via token replication. (We do not consider this model here.) 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 proofs. Improving upon their work, Goyal et al. [29] show a black-box construction of oblivious transfer based on stateless tokens. However, their protocol requires the parties to exchange $\Theta(\lambda)$ tokens for *every* oblivious transfer the parties wish to execute. Moreover, a flaw in their protocol has recently been identified [32].

A second direction has explored the possibility of eliminating computational assumptions altogether. This line of work was initiated by Moran and Segev [47], who showed how to realize statistically secure UC commitments using a single stateful token. (We remark that statistically secure UC commitments do not imply UC oblivious transfer [45].) Their construction can be used for any bounded number of commitments 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. [29] show an unconditional construction of oblivious transfer (and hence general secure computation) using $\Theta(\lambda)$ stateful tokens. Döttling et al. [20] show how to construct unconditionally secure OT using only a single stateful token. Goyal et al. [28] showed that unconditional security from stateless tokens is impossible unless the token model is extended to allow

	Here	Here	[13]	[32]
Tokens:	2	2	2	$\Theta(\lambda)$
Rounds:	$\Theta(1)$	$\Theta(1)$	$\Theta(\lambda)$	$\Theta(1)$
Assumption:	CRHF, VRF	CRHF	eTDP	OWF
Black box?	Yes	Yes	No	Yes
# OTs:	unbounded	bounded	unbounded	unbounded

Table 1: Universally composable OT based on stateless tamper-proof hardware tokens. The security parameter is denoted by λ. Reference [32] is subsequent work.

tokens to encapsulate each other (something we do not consider here). In that case, they show how to realize statistically secure OT in constant rounds using $\Theta(\lambda)$ stateless tokens.

Kolesnikov [42] showed an efficient construction of oblivious transfer from stateless tokens. However, that work does not give universally composable OT, and achieves only covert security [2] rather than security against malicious parties. Dubovitskaya et al. [22] constructed an OT protocol from two stateful tokens. Each pair of tokens, however, can be used for only a single OT.

Our work in relation to prior work. We show two efficient protocols for universally composable OT based on tamper-proof hardware tokens. Both our protocols use two *stateless* tokens (one generated by each party) and run in constant rounds. Our first protocol can be used for an unbounded number of OTs based on a single pair of exchanged tokens; this protocol assumes collision-resistant hash functions (CRHFs) and the existence of unique signatures [27] or, equivalently, verifiable random functions (VRFs) [46]. Our second protocol requires a known upper bound on the number of OTs to be carried out, but can be based on CRHFs alone. (OT extension [36] can be used to extend these to an unbounded number of OTs, however this appears to require additional assumptions.) A comparison of our protocols to other relevant 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 coin tossing (and hence OT) *can* be based on a single stateless token using nonblack-box techniques.

Concurrent and subsequent work. Concurrently and independently, Döttling et al. [21] showed a different nonblack-box construction of UC coin-tossing from a single stateless token, and argue (without proof) that nonblack-box techniques are needed. Here, we provide a rigorous version of their argument. Our efficient, black-box OT protocols using two stateless tokens—which we consider our primary contribution—have no counterpart in their work.

Subsequent to our work, Hazay et al. [32] showed a constant-round protocol for UC computation of any functionality using stateless tokens, assuming only one-way functions. However, their protocol requires $\Theta(\lambda)$ tokens.

Open questions. The main open question left by our work is whether there is a black-box construction of a *constant-round* protocol for an *unbounded* number of universally composable OTs assuming symmetric-key primitives (e.g., one-way functions and CRHFs) only.

2 Background and Cryptographic Primitives

We let λ denote the security parameter, and let PPT stand for "probabilistic polynomial-time." We use standard notions of security [40] for pseudorandom functions (denoted by PRF), collision-resistant hash functions, canonical message authentication codes (denoted by Mac), and digital signature schemes (denoted by $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$), though adapted for non-uniform polynomial-time adversaries. A signature scheme is called *unique* if for every possible public key vk and every message m, there is at most one signature σ such that $\mathsf{Vrfy}_{vk}(m, \sigma) = 1$. Various constructions of such schemes based on number-theoretic assumptions are known [46, 19, 1, 35, 33].

Let U_ℓ denote the uniform distribution on ℓ-bit strings. 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.

A *commitment scheme* $(\mathsf{Com}, \mathsf{Open})$ is a two-phase protocol executed by a sender and a receiver. In the commitment phase, which we assume is non-interactive, the sender holds an input z and random coins r; it generates a commitment com. In the opening phase, the sender sends z and r to the receiver, who verifies correctness by checking if $\mathsf{Open}(\mathsf{com}, z, r) = 1$. A scheme is *statistically binding* if it is binding for an all-powerful sender and hiding for a computationally bounded receiver, and *statistically hiding* if it is hiding for an all-powerful receiver and binding for a computationally bounded sender; we refer to [24] for the standard definitions. Statistically binding commitments can be constructed from any one-way function [48], while statistically hiding commitments can be based on collision-resistant hash functions [30, 17]. (In both these cases, the commitment phase requires two rounds; however, the first round can be done "once and for all" during pre-processing, after which any number of subsequent commitments can be done non-interactively.)

2.1 Linear Algebra

We briefly review some linear-algebraic facts used in our analysis. Let $\mathbb{F}_2$ denote the finite field with two elements. By default, all vectors are column vectors. If $a \in \mathbb{F}_2^\lambda$ then a^T denotes the transpose of a, which is a row vector.

Let $C \in \mathbb{F}_2^{n \times m}$ be a matrix of rank r. Then $\ker(C) = \{x \mid Cx = 0\}$ has dimension $m - r$. Letting $v_1, \ldots, v_{m-r} \in \mathbb{F}^m$ be a basis for $\ker(C)$, we can extend it to a basis $v_1, \ldots, v_m$ for $\mathbb{F}^m$. Let $e_i \in \mathbb{F}^{m-r}$ be the ith unit vector. There is a matrix $G \in \mathbb{F}_2^{(m-r) \times m}$ such that $Gv_i = e_i$ for $1 \le i \le m-r$ and $Gv_i = 0$ otherwise. We call any such matrix G a *complementary matrix* of C, and let $\mathsf{Comp}(C)$ denote some canonical way of computing G from C. Let $C \in \mathbb{F}_2^{n \times m}$ have rank $r = n$ and let G be a complementary matrix of C. If $v \in \mathbb{F}_2^m$ is uniform, then $Cv \in \mathbb{F}_2^n$ and $Gv \in \mathbb{F}_2^{m-n}$ are uniform and independent.

2.2 Ideal Functionalities

Token functionality. We model a tamper-proof hardware token as an ideal functionality in the UC framework, following Katz [39]; 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 functionalities. The OT functionality is standard, but we wish here to model a *multi*-session variant where the sender and receiver repeatedly (in sequential sub-sessions) execute some agreed-upon number m of parallel OTs (in a given sub-session). We refer to this functionality as $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$; see Figure 2. As highlighted in [29], the sender is notified when the receiver obtains output.

We define the bounded OT functionality in Figure 3 similarly except that the sender and receiver only execute a *single* session of m parallel OTs. Using OT pre-processing, this allows the sender and receiver to execute m sequential OTs with adaptively chosen inputs.

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 $(\textsc{create}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle, M)$ from a party C (i.e., the token creator), where U is another party (i.e., the token user) and 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 $\langle \mathsf{C}, \mathsf{U}, M, 0, \emptyset \rangle$. Send $(\textsc{create}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ to the adversary.

Deliver: Upon receiving $(\textsc{ready}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ from the adversary, send $(\textsc{ready}, \langle \mathsf{sid}, \mathsf{C}, \mathsf{U} \rangle)$ to U.

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

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

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

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

$\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ interacts with sender S, receiver R, and the adversary, and is parameterized by a security parameter λ and integer m. It also maintains variables $\mathsf{ready}_{\mathsf{S}}, \mathsf{ready}_{\mathsf{R}}$ initialized to $\perp$.

Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{(x_i^0, x_i^1)\}_{i=1}^m)$ from S, with $x_i^0, x_i^1 \in \{0,1\}^\lambda$, if $\mathsf{ready}_{\mathsf{S}} \neq \perp$ then ignore it. Otherwise, set $\mathsf{ready}_{\mathsf{S}} := \mathsf{ssid}$, record $\langle \mathsf{ssid}, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle$, and send $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to the adversary.

Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{b_i\}_{i=1}^m)$ from R, with $b_i \in \{0,1\}$, if $\mathsf{ready}_{\mathsf{R}} \neq \perp$ then ignore it. Otherwise, set $\mathsf{ready}_{\mathsf{R}} := \mathsf{ssid}$, record $\langle \mathsf{ssid}, \{b_i\}_{i=1}^m \rangle$, and send $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to the adversary.

Once $\langle \mathsf{ssid}, \{(x_i^0, x_i^1)\}_{i=1}^m \rangle$ and $\langle \mathsf{ssid}, \{b_i\}_{i=1}^m \rangle$ are recorded, for $\mathsf{ready}_{\mathsf{S}} = \mathsf{ready}_{\mathsf{R}} = \mathsf{ssid}$, send $\{x_i^{b_i}\}_{i=1}^m$ to R and $(\textsc{received}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid})$ to S, and set $\mathsf{ready}_{\mathsf{S}}, \mathsf{ready}_{\mathsf{R}} := \perp$.

Figure 2: The $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ functionality.

3 Oblivious Transfer Using Two Stateless Tokens

We begin in Section 3.1 by describing the intuition behind our protocols. In Section 3.2 we describe a protocol that allows the parties to execute an unbounded number of OTs based on the exchange of a single pair of tokens. We prove security of this protocol in Section 3.3 based on the assumptions of collision-resistant hash functions and unique signatures. In Section 3.4 we show how that protocol can be modified so it can be based on collision-resistant hash functions alone, at the expense of requiring the parties to fix a bound in advance on the number of OTs they can execute.

Functionality $\mathcal{F}_{\textsf{bounded-OT}}$

$\mathcal{F}_{\textsf{bounded-OT}}$ interacts with sender S, receiver R, and the adversary, and is parameterized by a security parameter λ and integer m.

Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \{(x_i^0, x_i^1)\}_{i=1}^m)$ from S with $x_i^0, x_i^1 \in \{0,1\}^\lambda$, record $\{(x_i^0, x_i^1)\}_{i=1}^m$, and send $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to the adversary. Ignore further $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \ldots)$ messages.

Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \{b_i\}_{i=1}^m)$, record $\{b_i\}_{i=1}^m$ and send $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to the adversary. Ignore further $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \ldots)$ messages.

Once $(\textsc{send}, \ldots)$ and $(\textsc{receive}, \ldots)$ messages have been received, send $\{x_i^{b_i}\}_{i=1}^m$ to R and $(\textsc{received}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle)$ to S.

Figure 3: The $\mathcal{F}_{\textsf{bounded-OT}}$ functionality.

3.1 Intuition and Background

Our starting point is the unconditionally secure OT protocol from [20], 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 $b \in \{0,1\}$. We canonically identify the vector space $\mathbb{F}_2^\lambda$ with $\{0,1\}^\lambda$. The main steps of the protocol are as follows:

1. S chooses a uniform $a \in \mathbb{F}_2^{2\lambda}$ and $B \in \mathbb{F}_2^{2\lambda \times 2\lambda}$. It then creates a token $\mathcal{T}_{\mathsf{S}}$ that, on input $z \in \mathbb{F}_2^{2\lambda}$, outputs $az^T + B$ and then refuses to answer any more queries. S sends $\mathcal{T}_{\mathsf{S}}$ to R.

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

3. R chooses uniform $z, h \in \mathbb{F}_2^{2\lambda}$ such that $z^T h = b$. It queries z to $\mathcal{T}_{\mathsf{S}}$, which outputs $V := az^T + B$. Then R checks that $CV = \tilde{a}z^T + \tilde{B}$, and aborts if not. Otherwise, R sends h to S.

4. S responds with $\tilde{x}_0 := x_0 + GBh$ and $\tilde{x}_1 := x_1 + GBh + Ga$, where $G = \mathsf{Comp}(C)$. The receiver then outputs $x_b := \tilde{x}_b - GVh$.

Correctness holds in an honest execution since $CV = C \cdot (az^T + B) = Caz^T + CB = \tilde{a}z^T + \tilde{B}$ and

$$\begin{aligned} \tilde{x}_b - GVh &= \big(x_b + GBh + Gab\big) - G \cdot (az^T + B) \cdot h \\ &= x_b + Gab - Gaz^T h \;=\; x_b, \end{aligned}$$

where we view $b \in \{0,1\}$ as an element of $\mathbb{F}_2^{1 \times 1}$.

Security can be argued, informally, as follows. For an honest receiver, the values z, h serve as a secret sharing of its input b, thus hiding b from S. To ensure correct behavior by S, including the way the token is implemented, the receiver checks that $CV = \tilde{a}z^T + \tilde{B}$. Since the behavior of the token cannot depend on the random matrix C, and $\tilde{a}, \tilde{B}$ cannot depend on z, incorrect behavior is detected with overwhelming probability.

As for an honest sender, note that $B' \overset{\text{def}}{=} GB$ and $a' \overset{\text{def}}{=} Ga$ are uniform conditioned on CB and Ca. Set $z_1 = z$, where z is the query made by R to the token, and extend this to a basis $\{z_1, \ldots, z_{2\lambda}\}$. Write the first row of B' as $\sum_i b_i' z_i^T$ with $b_i' \in \mathbb{F}_2$. From its query to the token, R can compute $GV = B' + a'z_1^T$ and hence the projections $b_2', \ldots, b_{2\lambda}'$ of the first row of B' onto each of $z_2^T, \dots, z_{2\lambda}^T$; however, it learns only $b'_1 + a'_1$ (where a'_1 denotes the first entry of a'). A similar argument holds for the other rows of B'. Now, the sender's inputs x_0, x_1 are "masked" by $B'h$ and $B'h + a'$, respectively. If $z_1^T h = 0$, then R can determine $B'h$ (since the projections of the rows of B' onto z_1^T are irrelevant), but $B'h + a'$ is uniform because a' is. On the other hand, if $z_1^T h = 1$ then R can deduce $B'h + a'$ (since it knows $B'z_1 + a'$), but $B'h$ is uniform (since the projections of the rows of B' onto z_1^T are uniform).

It is crucial that a malicious receiver can only query the token *once*. In the work of [20], this is enforced by making the token *stateful*; the token "self destructs" after the first query.

Extension to stateless tokens. We combine the techniques of [20] with ideas from [29] to design a protocol using two *stateless* tokens instead of one stateful token. We describe some of the difficulties this entails, though caution that this is only intuition and does not fully capture all the components of our protocol as specified in the following section:

Multiple queries. The main issue is preventing a malicious R from querying the token multiple times per OT. Motivated by similar techniques in [29], we address this issue by modifying the token so that it only replies to *authenticated* inputs. That is, rather than accepting an input z as before, the token now only responds to queries of the form $(\mathsf{com}_z, z, r_z, \sigma_z)$, where com_z is a commitment to z using randomness r_z, and σ_z is a signature on com_z with respect to a verification key vk_{S} of the sender.

Extracting the sender's input. With the above change in place, we are faced with a technical problem during simulation. Namely, even the simulator will now be unable to submit two (valid) inputs to the token in order to extract the sender's input values. To resolve this issue we introduce a second, stateless token $\mathcal{T}_{\mathsf{R}}$ sent from the receiver to the sender that takes as input a, B and returns $\tilde{a} = Ca$ and $\tilde{B} = CB$. This allows the simulator to extract a and B and thus to later extract the sender's inputs. As above, to prevent the sender from querying this token multiple times we modify the token so that it only responds to inputs that have been authenticated by the receiver.

We also modify both tokens so they output signatures serving as "proof" to the creator of the token that the token was queried (on a legal input) before a certain point in the protocol.

Malicious tokens. A malicious token may try to leak information to the token creator about the queries that were submitted to it. In particular, such leakage could potentially be embedded in the signature output by a token. To protect against this, we require the underlying signature scheme to be *unique*.

Using unique signatures ensures that the only information leakage that can occur is due to a *token abort*. Note that such an abort might reveal information about all previous executions, though it can occur at most once (since an abort is evidence of cheating). It turns out that for $\mathcal{T}_{\mathsf{S}}$, leakage due to an abort is already handled by the protocol of [20]. For $\mathcal{T}_{\mathsf{R}}$, we rely (intuitively) on the fact that an abort leaks only a limited amount of information.

Multiple executions. To achieve an unbounded number of OTs using a single pair of tokens we replace all random values with values output by a pseudorandom function.

```
On input $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$:
 if $\big(\mathsf{Vrfy}_{vk_\mathsf{S}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_z, \sigma_z) = 1$
    and $\mathsf{Open}(\mathsf{com}_z, z, r_z) = 1\big)$
      $a := \mathsf{PRF}_{k_a}(\mathsf{ssid}\|i)$
      $B := \mathsf{PRF}_{k_B}(\mathsf{ssid}\|i)$
      $V := az^T + B$
      $\sigma := \mathsf{Sign}_{sk_\mathsf{S}}(\mathsf{ssid}\|i\|1)$
      output $(V, \sigma)$
 else output $\bot$
```

Figure 4: The Turing machine M_S to be embedded in the sender-created token $\mathcal{T}_\mathsf{S}$.

```
On input $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$:
 if $\big(\mathsf{Vrfy}_{vk_\mathsf{R}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{a\|B}, \sigma_{a\|B}) = 1$
    and $\mathsf{Open}(\mathsf{com}_{a\|B}, a\|B, r_{a\|B}) = 1\big)$
      $C := \mathsf{PRF}_{k_C}(\mathsf{ssid})$
      $\tilde{a} := Ca$
      $\tilde{B} := CB$
      $\sigma_{\tilde{a}\|\tilde{B}} := \mathsf{Sign}_{sk_\mathsf{R}}(\mathsf{ssid}\|i\|1\|\tilde{a}\|\tilde{B})$
      output $(\tilde{a}, \tilde{B}, \sigma_{\tilde{a}\|\tilde{B}})$
 else output $\bot$
```

Figure 5: The Turing machine M_R to be embedded in the receiver-created token $\mathcal{T}_\mathsf{R}$.

3.2 The Protocol

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a unique signature scheme, PRF be a pseudorandom function, Ext be a strong extractor (with parameters fixed later), and $(\mathsf{SCom}, \mathsf{Open})$ be a non-interactive, statistically hiding commitment scheme. Our main protocol π consists of a token-exchange phase after which the sender S and receiver R can carry out an unlimited number of OTs. A formal description follows; see also Figure 6.

Token-exchange phase. Each party generates a single token and sends it to the other party. The sender chooses uniform $k_a, k_B \in \{0,1\}^\lambda$ and computes $(sk_\mathsf{S}, vk_\mathsf{S}) \leftarrow \mathsf{Gen}(1^\lambda)$; it then creates a token $\mathcal{T}_\mathsf{S}$ containing the code from Figure 4. The receiver chooses uniform $k_C \in \{0,1\}^\lambda$ and computes $(sk_\mathsf{R}, vk_\mathsf{R}) \leftarrow \mathsf{Gen}(1^\lambda)$; it then creates a token $\mathcal{T}_\mathsf{R}$ containing the code from Figure 5. The parties then exchange their tokens as well as their public keys $vk_\mathsf{S}, vk_\mathsf{R}$. (Each party also runs the first round of a two-round, statistically hiding commitment scheme; we leave this implicit.)

Oblivious-transfer phase. Following the above, the parties can sequentially run an unbounded number of sub-sessions, where in each sub-session they can carry out any desired number m of OTs in parallel. (If a party ever aborts the protocol during some sub-session, however, the other party refuses to run any subsequent sub-sessions.) The protocol run during each sub-session, where the sender S has input $\{(x_i^0, x_i^1)\}_{i=1}^m$ and the receiver R has input $\{b_i\}_{i=1}^m$, proceeds as follows:

Step 1: For $i \in [m]$, the sender computes $a_i := \mathsf{PRF}_{k_a}(\mathsf{ssid}\|i)$ and $B_i := \mathsf{PRF}_{k_B}(\mathsf{ssid}\|i)$, where $a_i \in \mathbb{F}_2^{4\lambda}$ and $B_i \in \mathbb{F}_2^{4\lambda\times 4\lambda}$. The sender then commits to each (a_i, B_i) using SCom, resulting in commitment $\mathsf{com}_{a_i\|B_i}$ and decommitment $r_{a_i\|B_i}$. It sends $\{\mathsf{com}_{a_i\|B_i}\}_{i=1}^m$ to R.

Step 2: For $i \in [m]$ the receiver chooses uniform $h_i, z_i \in \mathbb{F}_2^{4\lambda}$ subject to $b_i = z_i^T h_i$. It commits to each z_i using SCom, resulting in commitment com_{z_i} and decommitment r_{z_i}. The receiver next computes $C := \mathsf{PRF}_{k_C}(\mathsf{ssid})$, where $C \in \mathbb{F}_2^{2\lambda\times 4\lambda}$, and $\sigma_{a_i\|B_i} := \mathsf{Sign}_{sk_\mathsf{R}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{a_i\|B_i})$ for all i. It sends C, $\{\mathsf{com}_{z_i}\}_{i=1}^m$, and $\{\sigma_{a_i\|B_i}\}_{i=1}^m$ to S.

Step 3: S verifies the signatures just received, and aborts if any of them is invalid. Otherwise, for all i the sender runs the token $\mathcal{T}_\mathsf{R}$ on input $(\mathsf{ssid}, i, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \sigma_{a_i\|B_i})$, and obtains in return $(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i})$. It verifies that $\tilde{a}_i = Ca_i$ and $\tilde{B}_i = CB_i$ for all i, and that the signatures output by $\mathcal{T}_\mathsf{R}$ are valid, and aborts if not. Otherwise, S computes $\sigma_{z_i} := \mathsf{Sign}_{sk_\mathsf{S}}(\mathsf{ssid}\|i\|\mathsf{com}_{z_i})$ for all i, and sends $\{(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}, \sigma_{z_i})\}_{i=1}^m$ to R.

Step 4: The receiver verifies the signatures just received, and aborts if any of them is invalid. Otherwise, for all i the receiver runs the token $\mathcal{T}_{\mathsf{S}}$ on input $(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \sigma_{z_i})$ and obtains in return (V_i, σ_i), where σ_i is a signature on sid, i. It verifies that $CV_i = \tilde{a}_i z_i^T + \tilde{B}_i$ for all i, and that the signatures output by $\mathcal{T}_{\mathsf{S}}$ are valid, and aborts if not. Otherwise, it sends $\{(h_i, \sigma_i)\}_{i=1}^m$ to S.

Step 5: The sender verifies the signatures just received, and aborts if any of them is invalid. Otherwise, S computes $G := \mathsf{Comp}(C)$, and then for all i chooses uniform extractor keys v_i^0, v_i^1 and computes $\tilde{x}_i^0 := \mathsf{Ext}(GB_i h_i, v_i^0) \oplus x_i^0$ and $\tilde{x}_i^1 := \mathsf{Ext}(GB_i h_i + Ga_i, v_i^1) \oplus x_i^1$. Finally, it sends $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m$ to the receiver.

Step 6 (output determination): The receiver computes $G := \mathsf{Comp}(C)$ and then, for $i \in [m]$, outputs $x_i^{b_i} := \tilde{x}_i^{b_i} \oplus \mathsf{Ext}(GV_i h_i, v_i^{b_i})$.

We sketch the intuition for why the protocol is secure.

Corrupted sender. Consider what a malicious sender S^* sees as part of an execution of the oblivious-transfer phase during some sub-session. S^* learns C and $\{h_i\}$, which—as in the protocol of [20]—do not leak information about the receiver's input bits $\{b_i\}$. The signatures $\{\sigma_{a_i \| B_i}\}$ computed by the receiver are independent of the $\{b_i\}$. The only other information S^* learns are the signatures $\{\sigma_i\}$ computed by the (possible misbehaving) token $\mathcal{T}_{\mathsf{S}^*}$. Since a unique signature scheme is used, however, these signatures are uniquely determined by values that are independent of the $\{b_i\}$, and so $\mathcal{T}_{\mathsf{S}^*}$ cannot communicate information about the $\{b_i\}$ back to S^*.

The preceding statement is true except for one subtlety: $\mathcal{T}_{\mathsf{S}^*}$ can potentially communicate information to S^* by *aborting* (i.e., refusing to output a valid signature) at some point in the protocol. Note further that such an abort can, in general, depend on all *previous* oblivious-transfer phases that have been executed. Nevertheless, because the token can only abort *once* (after which point the receiver refuses to run any subsequent executions), and there are polynomially many executions overall, this strategy can be used to leak only $O(\log \lambda)$ bits of information to S^*. Moreover, this leakage cannot depend on the $\{h_i\}$. (Here we rely on the fact that the $\{\sigma_{z_i}\}$ cannot convey additional information from S^* to $\mathcal{T}_{\mathsf{S}^*}$, again because a unique signature scheme is used.) We show that even with this leakage, S^* gets only negligible information on the $\{b_i\}$ of any sub-session. Roughly speaking, this is a consequence of the fact that $b_i = z_i^T h_i$ is statistically close to uniform, even conditioned on h_i, as long as z_i has sufficient min-entropy.

Looking ahead, as part of the proof of security we will need to show how a simulator can *extract* the effective inputs of S^*. This will be done as follows. For each sub-session ssid and index i, the malicious sender must query the token $\mathcal{T}_{\mathsf{R}}$ on some valid input or else the receiver will abort in step 4; this follows from unforgeability of the signature scheme. Moreover, for any pair (ssid, i), the malicious sender can only (usefully) query $\mathcal{T}_{\mathsf{R}}$ on inputs $(\mathsf{ssid}, i, \mathsf{com}_{a_i \| B_i}, a_i, B_i, r_{a_i \| B_i}, \sigma_{a_i \| B_i})$ for a *single* pair a_i, B_i; this follows from statistical binding of the commitment scheme and unforgeability of the signature scheme. From this (unique) query, the simulator can determine a_i, B_i and use those to define the sender's effective inputs.

Corrupted receiver. We show that a malicious receiver R^* learns only one of the sender's inputs in each OT carried out in some sub-session. During an execution, R^* learns $\{(\tilde{a}_i, \tilde{B}_i)\}$ and $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}$, which—as in the protocol of [20]—only enables R^* to learn one of the sender's inputs per OT. (Here the sender's inputs are masked by strings that are *extracted from* $GB_i h_i$ and $GB_i h_1 + Ga_i$, rather than being masked by those values themselves, but this is inconsequential.)

```
S                                                         R

Exchange tokens and public keys:
$k_a, k_B \leftarrow \{0,1\}^\lambda; (sk_\mathsf{S}, vk_\mathsf{S}) \leftarrow \mathsf{Gen}(1^\lambda)$
generate token $\mathcal{T}_\mathsf{S}$ as in Figure 4
                                                          $k_C \leftarrow \{0,1\}^\lambda; (sk_\mathsf{R}, vk_\mathsf{R}) \leftarrow \mathsf{Gen}(1^\lambda)$
                                                          generate token $\mathcal{T}_\mathsf{R}$ as in Figure 5

            $\xleftrightarrow{\mathcal{T}_\mathsf{S}, vk_\mathsf{S}; \mathcal{T}_\mathsf{R}, vk_\mathsf{R}}$

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

$(\mathsf{ssid}, m, \{(x_i^0, x_i^1)\}_{i=1}^m)$          $(\mathsf{ssid}, m, \{b_i\}_{i=1}^m)$
For $i \in [m]$ :
    $a_i := \mathsf{PRF}_{k_a}(\mathsf{ssid}\|i)$
    $B_i := \mathsf{PRF}_{k_B}(\mathsf{ssid}\|i)$
    $\mathsf{com}_{a_i\|B_i} \leftarrow \mathsf{SCom}(a_i\|B_i; r_{a_i\|B_i})$

            $\xrightarrow{\{\mathsf{com}_{a_i\|B_i}\}_{i=1}^m}$

                                                          $C := \mathsf{PRF}_{k_C}(\mathsf{ssid})$
                                                          For $i \in [m]$ :
                                                              $h_i \leftarrow \mathbb{F}_2^{4\lambda} \setminus \{0^{4\lambda}\}$
                                                              $z_i \leftarrow \{z_i \in \mathbb{F}_2^{4\lambda} \mid z_i^T h_i = b_i\}$
                                                              $\mathsf{com}_{z_i} \leftarrow \mathsf{SCom}(z_i; r_{z_i})$
                                                              $\sigma_{a_i\|B_i} := \mathsf{Sign}_{sk_\mathsf{R}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{a_i\|B_i})$

            $\xleftarrow{C, \{(\mathsf{com}_{z_i}, \sigma_{a_i\|B_i})\}_{i=1}^m}$

For $i \in [m]$ :
    If $\mathsf{Vrfy}_{vk_\mathsf{R}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{a_i\|B_i}, \sigma_{a_i\|B_i}) \neq 1$
        abort
    Run $\mathcal{T}_\mathsf{R}\left(\mathsf{ssid}, i, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \sigma_{a_i\|B_i}\right)$
        to get $(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i})$
    If $\tilde{a}_i \neq Ca_i$ or $\tilde{B}_i \neq CB_i,$ abort
    If $\mathsf{Vrfy}_{vk_\mathsf{R}}(\mathsf{ssid}\|i\|1\|\tilde{a}_i\|\tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}) \neq 1$
        abort
    $\sigma_{z_i} := \mathsf{Sign}_{sk_\mathsf{S}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{z_i})$

            $\xrightarrow{\{(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}, \sigma_{z_i})\}_{i=1}^m}$

                                                          For $i \in [m]$ :
                                                              If one of the following fails, abort:
                                                                  $\mathsf{Vrfy}_{vk_\mathsf{R}}(\mathsf{ssid}\|i\|1\|\tilde{a}_i\|\tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i})$
                                                                  $\mathsf{Vrfy}_{vk_\mathsf{S}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{z_i}, \sigma_{z_i})$
                                                              Run $\mathcal{T}_\mathsf{S}(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \sigma_{z_i})$
                                                                  to get $(V_i, \sigma_i)$
                                                              If $\Big(\mathsf{Vrfy}_{vk_\mathsf{S}}(\mathsf{ssid}\|i\|1, \sigma_i) \neq 1$
                                                                  or $CV_i \neq \tilde{a}_i z_i^T + \tilde{B}_i\Big)$ abort

            $\xleftarrow{\{(h_i, \sigma_i)\}_{i=1}^m}$

For $i \in [m]$ :
    If $\mathsf{Vrfy}_{vk_\mathsf{S}}(\mathsf{ssid}\|i\|1, \sigma_i) \neq 1,$ then abort
$G := \mathsf{Comp}(C)$
For $i \in [m]$ :
    choose keys $v_i^0, v_i^1$
    $\tilde{x}_i^0 := \mathsf{Ext}(GB_i h_i, v_i^0) \oplus x_i^0$
    $\tilde{x}_i^1 := \mathsf{Ext}(GB_i h_i + Ga_i, v_i^1) \oplus x_i^1$

            $\xrightarrow{\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m}$

                                                          $G := \mathsf{Comp}(C)$
                                                          For $i \in [m]$ :
                                                              output $\tilde{x}_i^{b_i} \oplus \mathsf{Ext}(GV_i h_i, v_i^{b_i})$
```

Figure 6: An OT protocol π from two stateless tokens. The token-exchange phase is run once; the second phase can be run any number of times.

R^* also sees commitments to the $\{a_i, B_i\}$, as well as signatures $\{\sigma_{z_i}\}$ computed by the sender, but these do not leak any additional information on the sender's inputs. The only other values R^* learns are the signatures $\{\sigma_{\tilde{a}_i \| \tilde{B}_i}\}$. However, the fact that a unique signature scheme is used means that these signatures are uniquely determined by values that R^* already knows, and so there is no way for the (possibly misbehaving) token $\mathcal{T}_{\mathsf{R}^*}$ to communicate information to R^*.

As in the case of a corrupted sender, the preceding statement is true except that $\mathcal{T}_{\mathsf{R}^*}$ can leak $O(\log \lambda)$ bits of information to R^* by aborting at some point, possibly in a subsequent sub-session. It is for this reason that we apply a strong extractor to GB_ih_i and $GB_ih_i + Ga_i$, rather than using those values directly as masks for the sender's inputs.

In the formal proof we will need to show how a simulator can extract the effective inputs of R^*. This is done by again observing that for each sub-session ssid and index i, the malicious receiver can (usefully) query the token $\mathcal{T}_{\mathsf{S}}$ on inputs $(\mathsf{ssid}, i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \sigma_{z_i})$ for only a single z_i. With z_i and h_i known, the simulator can extract the malicious receiver's effective input.

3.3 Proof of Security

Theorem 1 *If* PRF *is a pseudorandom function,* SCom *is a statistically hiding commitment scheme,* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a unique signature scheme, and* Ext *is a* $(1.5\lambda, \mathsf{negl}(\lambda))$*-strong extractor, then the protocol of Figure 6 securely realizes* $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ *in the* $\mathcal{F}_{\mathsf{wrap}}$*-hybrid model.*

To prove the theorem, we construct a straight-line simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish between (1) an execution involving an honest party and a corrupted party (that without loss of generality we may take as the dummy adversary who simply forwards messages to/from $\mathcal{Z}$) running protocol π in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving the same honest party, functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ and Sim in the ideal world. We consider the cases of a corrupted sender and a corrupted receiver separately.

3.3.1 Corrupted Sender

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest receiver running π and the dummy adversary $\mathcal{A}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest receiver, functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$, and Sim in the ideal world. Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{A}$.

Tokens. Sim simulates $\mathcal{A}$'s receipt of a token from the honest receiver (formally, via the $\mathcal{F}_{\mathsf{wrap}}$ functionality) by computing $(sk_{\mathsf{R}}, vk_{\mathsf{R}}) \leftarrow \mathsf{Gen}(1^\lambda)$ and giving $(\textsc{ready}, \langle \mathsf{sid}, \mathsf{R}, \mathsf{S} \rangle)$ and vk_{R} to $\mathcal{A}$. It then waits to receive vk_{S} as well as a message $(\textsc{create}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, M_{\mathcal{A}})$ from $\mathcal{A}$ to the $\mathcal{F}_{\mathsf{wrap}}$ functionality, and records $M_{\mathcal{A}}$. For readability, we let $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ (resp., $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}} = M_{\mathcal{A}}$) denote the $\mathcal{F}_{\mathsf{wrap}}$ functionality corresponding to the receiver's (resp., sender's) token.

When $\mathcal{A}$ queries $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$, the simulator answers the query as in Figure 5, except that a random function is used in place of $\mathsf{PRF}_{k_C}(\cdot)$. (I.e., Sim dynamically maintains a table, answering new queries with random values but answering consistently if the same query is asked twice.) A query $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$ to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ is *valid* if $\mathsf{Vrfy}_{vk_{\mathsf{R}}}(\mathsf{ssid}\|i\|0\|\mathsf{com}_{a\|B}, \sigma_{a\|B}) = 1$ and $\mathsf{Open}(\mathsf{com}_{a\|B}, a\|B, r_{a\|B}) = 1$. When a valid query is answered with $(\tilde{a}, \tilde{B}, \sigma_{\tilde{a}\|\tilde{B}})$, the simulator records (ssid, i, a, B). If, at any point $\mathcal{A}$ makes a valid query $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$ to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ but $\sigma_{a\|B}$ was not previously generated by the simulator as a signature on $\mathsf{ssid}\|i\|0\|\mathsf{com}_{a\|B}$, or $\mathcal{A}$ makes two valid queries $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$ and $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a', B', r'_{a\|B}, \sigma_{a\|B})$ with $(a, B) \neq (a', B')$, the simulator aborts.

Simulation of the execution for sub-session ssid. The simulator Sim proceeds as follows:

1. Receive commitments $\{\mathsf{com}_{a_i\|B_i}\}$ in step 1 of the protocol.

2. In step 2, a random function is used in place of $\mathsf{PRF}_{k_C}(\cdot)$ (as discussed above). Sim also chooses h_i, z_i subject to $z_i^T h_i = 0$ for all i, and runs the rest of this step honestly using these values. Let $\{\sigma_{a_i\|B_i}\}$ be the signatures sent to $\mathcal{A}$.

3. In step 4, after receiving $\{(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}, \sigma_{z_i})\}_{i=1}^m$ from $\mathcal{A}$, the simulator runs the protocol honestly (including interacting with $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$), and aborts if dictated by the protocol. If it does not abort then for each i it checks for a record of the form $(\mathsf{ssid}, i, \star, \star)$. (Recall the simulator records (ssid, i, a, B) for every valid query to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$, and aborts if there are ever two valid queries for the same ssid, i with different a, B.) If there is no such record, or if there is a record $(\mathsf{ssid}, i, a_i, B_i)$ but $(\tilde{a}_i, \tilde{B}_i) \neq (Ca_i, CB_i)$, then Sim aborts.

   Assuming it did not abort, Sim sends the next message of the protocol exactly as the honest receiver would.

4. Upon receiving $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m$ from $\mathcal{A}$, for each i the simulator finds the unique record of the form $(\mathsf{ssid}, i, a_i, B_i)$. It then computes $x_i^0 := \tilde{x}_i^0 \oplus \mathsf{Ext}(GB_ih_i, v_i^0)$ and $x_i^1 := \tilde{x}_i^1 \oplus \mathsf{Ext}(GB_ih_i + Ga_i, v_i^1)$ for all i. Finally, it sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \{(x_i^0, x_i^1)\}_{i=1}^m)$ to functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$.

Let Real denote the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of π between an honest receiver and $\mathcal{A}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the outputs of the honest receiver.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest receiver, $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. In this experiment, we imagine an interaction between a simulator Sim' (playing the role of the sender), the honest receiver, and $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ in the ideal world, but where Sim' is given the inputs $\{b_i\}_{i=1}^m$ of the receiver at the outset of every sub-session. Sim' internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. It also simulates an execution of protocol π with $\mathcal{A}$, by playing the role of the receiver itself using the inputs $\{b_i\}_{i=1}^m$. As part of this simulation, Sim' emulates $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ and interacts with $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$. A query to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ is defined to be *valid* as above, and after each valid query $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$, the simulator records (ssid, i, a, B).

When execution of a sub-session of π is complete then, if it was not aborted, Sim' obtains outputs $\{x_i^{b_i}\}_{i=1}^m$. It sets $x_i^{\bar{b}_i} := 0^\lambda$ for $i \in [m]$ and sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \mathsf{ssid}, \{(x_i^0, x_i^1)\}_{i=1}^m)$ to the functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$.

It is immediate that the view of $\mathcal{Z}$ in $\mathcal{H}_1$ is distributed identically to its view in Real.

Experiment $\mathcal{H}_2$. This experiment is the same as the previous one, except that here Sim' uses a random function in place of $\mathsf{PRF}_{k_C}(\cdot)$. It is clear that if PRF is a pseudorandom function, then the view of $\mathcal{Z}$ here is computationally indistinguishable from its view in $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. This experiment is the same as the previous one except that if, at any point during the experiment, $\mathcal{A}$ makes a valid query $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ but $\sigma_{a\|B}$ was not previously generated by the simulator as a signature on $\mathsf{ssid}\|i\|0\|\mathsf{com}_{a\|B}$, the simulator aborts. It is easy to see that strong unforgeability of the signature scheme (which is implied by unforgeability and uniqueness of the signature scheme) means that the view of $\mathcal{Z}$ in this experiment is computationally indistinguishable from its view in $\mathcal{H}_2$.

Experiment $\mathcal{H}_4$. This experiment is the same as the previous one, except that if $\mathcal{A}$ ever makes two valid queries $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \sigma_{a\|B})$ and $(\mathsf{ssid}, i, \mathsf{com}_{a\|B}, a', B', r'_{a\|B}, \sigma_{a\|B})$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ with $(a, B) \neq (a', B')$, the simulator aborts. It is easy to see that (computational) binding of the commitment scheme implies that the view of $\mathcal{Z}$ here is indistinguishable from its view in $\mathcal{H}_3$.

Experiment $\mathcal{H}_5$. This is identical to the previous experiment, except that upon receiving a message $\{(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}, \sigma_{z_i})\}_{i=1}^m$ from $\mathcal{A}$ the simulator proceeds as follows: It first verifies the signatures and aborts if they are invalid (as in the previous experiment). Otherwise, for each i the simulator checks for a record of the form $(\mathsf{ssid}, i, \star, \star)$. If there is no such record, or if there is a record $(\mathsf{ssid}, i, a_i, B_i)$ but $(\tilde{a}_i, \tilde{B}_i) \neq (Ca_i, CB_i)$, then Sim' aborts. Since the output of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ on input $(\mathsf{ssid}, i, \ldots)$ includes Ca_i, CB_i, and a signature $\sigma_{\tilde{a}_i\|\tilde{B}_i}$ on those values, unforgeability of the signature scheme implies that the view of $\mathcal{Z}$ here is indistinguishable from its view in $\mathcal{H}_4$.

Experiment $\mathcal{H}_6$. This is identical to $\mathcal{H}_5$, except that in step 4 the simulator proceeds as follows (assuming it has not yet aborted): for each i, it finds the unique record of the form $(\mathsf{ssid}, i, a_i, B_i)$. Then it aborts if $V_i \neq a_i z_i^T + B_i$; otherwise, it proceeds as before. We claim that the view of $\mathcal{Z}$ in $\mathcal{H}_5$ is statistically close to its view in $\mathcal{H}_6$. To see this, recall that the simulator aborts in $\mathcal{H}_5$ and $\mathcal{H}_6$ if $\tilde{a}_i \neq Ca_i$ or $\tilde{B}_i \neq CB_i$ (where $\tilde{a}_i, \tilde{B}_i$ are the values received from $\mathcal{A}$ in step 4); in $\mathcal{H}_5$, the simulator then aborts if $CV_i \neq \tilde{a}_i z_i^T + \tilde{B}_i$ whereas in $\mathcal{H}_6$ it aborts if $V_i \neq a_i z_i^T + B_i$. Thus, the only difference between $\mathcal{H}_5$ and $\mathcal{H}_6$ occurs if there is ever a sub-session and an index i for which $CV_i = \tilde{a}_i z_i^T + \tilde{B}_i = C \cdot \left(a_i z_i^T + B_i\right)$ but $V_i \neq a_i z_i^T + B_i$. Intuitively, this occurs with negligible probability since $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ does not have any information about C. We now prove this formally.

We bound the probability of the event in question for some fixed ssid, i; a union bound then implies the claimed result. So, let E be the event that (1) the simulation is not aborted before the ith invocation of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ in sub-session ssid, and (2) the ith invocation of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ in that sub-session outputs V_i with $V_i \neq a_i z_i^T + B_i$ but $CV_i = C \cdot \left(a_i z_i^T + B_i\right)$, where $\mathcal{A}$ sent $(\tilde{a}_i, \tilde{B}_i, \sigma_{\tilde{a}_i\|\tilde{B}_i}, \sigma_{z_i})$ in step 4 of sub-session ssid. Let $\delta(\lambda)$ denote the probability of event E.

Consider running the entire experiment up to the ith invocation of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ in sub-session ssid and then rewinding and choosing a fresh, uniform value C' for that sub-session (keeping all other random choices the same as in the first run of the experiment). Using Jensen's inequality, we have that with probability at least $\delta(\lambda)^2$ event E occurs in both executions. Let a_i, B_i denote the values recorded by the simulator in the first execution, and let a'_i, B'_i denote the values recorded by the simulator in the second execution. Note that:

- The probability that $(a_i, B_i) \neq (a'_i, B'_i)$ is negligible, by straightforward reduction to the binding of the commitment scheme. (Note that the sender commits to a_i, B_i before learning C.)
- The probability that E occurs in both executions if $(a_i, B_i) = (a'_i, B'_i)$ is negligible. To see this, note that V_i must be the same in both executions, since the input to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ is the same in both executions. (Here we rely on uniqueness of the signature scheme.) So the probability that E occurs in both executions is at most the probability that the value C' in the second

execution satisfies $C' \cdot (V_i - a_i z_i^T - B_i) = 0$, where $V_i - a_i z_i^T - B_i \neq 0$. Since $C' \in \mathbb{F}_2^{2\lambda \times 4\lambda}$ is chosen uniformly and independently of everything else, this probability is at most $2^{-2\lambda}$.

We conclude that the probability that E occurs in both executions (i.e., $\delta(\lambda)^2$) is negligible; hence $\delta(\lambda)$ is negligible as well.

Experiment $\mathcal{H}_7$. Now, in step 6, Sim' finds the unique record of the form $(\mathsf{ssid}, i, a_i, B_i)$ for each i. The simulator then computes $x_i^0 := \tilde{x}_i^0 \oplus \mathsf{Ext}(GB_i h_i, v_i^0)$ and $x_i^1 := \tilde{x}_i^1 \oplus \mathsf{Ext}(Ga_i + GB_i h_i, v_i^1)$. The rest of its execution is as before.

We claim that the views of $\mathcal{Z}$ in $\mathcal{H}_6$ and $\mathcal{H}_7$ are identical. The change in the computation of $x_i^{\bar{b}_i}$ does not affect the view of $\mathcal{Z}$ at all (since the honest party's input is b_i). We consider next the change in the computation of $x_i^{b_i}$. In both $\mathcal{H}_6$ and $\mathcal{H}_7$, the simulator aborts unless $V_i = a_i z_i^T + B_i$. In step 6 of $\mathcal{H}_6$, the simulator computes $x_i^{b_i} := \tilde{x}_i^{b_i} \oplus \mathsf{Ext}(GV_i h_i, v_i^{b_i})$, where V_i is the value it received from $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ in step 4. But then in $\mathcal{H}_7$ we have

$$Ga_i b_i + GB_i h_i = G \cdot (a_i z_i^T + B_i) h_i = GV_i h_i,$$

and so the result computed for $x_i^{b_i}$ in $\mathcal{H}_7$ is the same.

Experiment $\mathcal{H}_8$. This is identical to the previous experiment, except that now Sim' chooses $z_i \leftarrow \{z \in \mathbb{F}_2^{4\lambda} \mid z^T h_i = 0\}$ for all i in every sub-session (rather than $z_i \leftarrow \{z \in \mathbb{F}_2^{4\lambda} \mid z^T h_i = b_i\}$). We prove that the distributions on the view of $\mathcal{Z}$ in $\mathcal{H}_7$ and $\mathcal{H}_8$ are statistically close. Note that this is not hard to prove if there is only a *single* execution of the protocol: in that case, the view of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ is independent of the $\{h_i\}$, and the view of $\mathcal{A}$ is independent of the $\{z_i\}$ except possibly for one bit of information that $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ can transmit by aborting. When we consider multiple sequential executions, however, the situation becomes more complicated since subsequent executions can (potentially) be used to transmit information between $\mathcal{A}$ and $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ about previous sub-sessions.

To analyze this possibility, we consider a sequence $\mathcal{H}_7^0, \mathcal{H}_7^1, \ldots$ of hybrid experiments between $\mathcal{H}_7$ and $\mathcal{H}_8$, where in going from one experiment to the next we change the distribution of a single z_i (i.e., for one index i in a single sub-session) while keeping the distributions of all the remaining z-values the same. We then show that we can construct algorithms A and T that interact in something we call *the dot-product-with-equality-oracle game*, such that the advantage of A in that game is related to the difference between the probabilities with which $\mathcal{Z}$ outputs 1 in two consecutive hybrid experiments. Finally, we show in Theorem 5 (see Appendix A) that the advantage of any A in the dot-product-with-equality-oracle game is negligible.

The dot-product-with-equality-oracle game is defined as follows: First, a uniform bit b is chosen. Values $z, h \in \mathbb{F}_2^{4\lambda}$ are then chosen uniformly subject to $z^T h = b$ and h nonzero; next, A is given h, and T is given z. Parties A and T then interact with a third party for at most $p(\lambda)$ rounds; in round j, they each send arbitrary inputs a_j and t_j to this third party, who responds to both parties with a bit indicating whether a_j and t_j are equal. If so, the parties continue to the next round; once $a_j \neq t_j$ the protocol ends. (Otherwise, the protocol ends in round $p(\lambda)$.) When the protocol ends, A outputs a guess for b and succeeds if its guess is correct. The advantage of A is the absolute value of its probability of success minus 1/2. We prove in Appendix A that for any (even all-powerful) A, T, and polynomial p, the advantage of A in this game is negligible.

We now show how to map the experiments in question to an execution of the above game. For concreteness, consider the case where the distribution of z_j in sub-session ℓ is changed. We define (unbounded) algorithms A and T as follows. At a high level, algorithms A and T will each encapsulate the entire experiment—including $\mathcal{A}, \mathsf{Sim}'$, the $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ functionality, the honest receiver, and $\mathcal{Z}$—using shared randomness, except that A uses its input h for the value of h_j in sub-session ℓ, and T uses its input z for the value of z_j in sub-session ℓ.

Explicitly, A runs the entire experiment internally. For sub-sessions 1 through $\ell-1$, it simulates the entire interaction between $\mathcal{A}$ and Sim' in the obvious way, including simulation of $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$. In sub-session ℓ, it simulates rounds 1–3 of the protocol, and then sends 1 to the (external) third party computing equality; if that party returns 0 then A simulates Sim' sending an abort for the fourth message of the protocol, passes the view of $\mathcal{A}$ to $\mathcal{Z}$, and outputs whatever $\mathcal{Z}$ outputs. Otherwise, A continues to the fourth round of the protocol, using values $\{h_i\}$ chosen in advance except that it sets h_j equal to its input h. Execution of the rest of sub-session ℓ continues in the natural way.

For sub-sessions $\ell+1$ and on, A simulates each sub-session in the natural way until after the third message of the protocol. It then computes $\{a_i, B_i\}_{i=1}^m$ as Sim' would, and sends $\{a_i z_i^T + B_i\}_{i=1}^m$ to the (external) third party computing equality. If that party returns 0 then A simulates Sim' sending an abort for the fourth message of the protocol, passes the view of $\mathcal{A}$ to $\mathcal{Z}$, and outputs what $\mathcal{Z}$ outputs. Otherwise, A continues simulation of the experiment until $\mathcal{Z}$ generates output.

We now describe algorithm T. As noted above, A and T share randomness in advance, and this allows T to simulate sub-sessions 1 through $\ell-1$ exactly as A does. In sub-session ℓ, it simulates rounds 1–3 of the protocol, and then queries $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ using $\{z_i\}$ chosen in advance except that z_j is set equal to its input z. (One subtlety is that A and T must agree on the commitment com_{z_j} used in sub-session ℓ even though they don't know z_j in advance. But since the commitment is statistically hiding and T is unbounded, T can decommit to $z_j = z$ except with negligible probability.) It receives in return $\{(V_i, \sigma_i)\}$, and checks whether $V_i = a_i z_i^T + B_i$ for all i; if so, it sends 1 to the (external) third party computing equality, and otherwise it sends 0.

For sub-sessions $\ell+1$ and on, T simulates each sub-session in the natural way until after the third message of the protocol. It then queries $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ and receives in return $\{(V_i, \sigma_i)\}_{i=1}^m$. Assuming the signatures verify, it sends $\{V_i\}_{i=1}^m$ to the (external) third party computing equality.

One can check that the interaction between A and T in the dot-product-with-equality-oracle game results in a perfect simulation of the entire experiment for $\mathcal{Z}$. (We highlight the fact that the experiment is independent of h and z until $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ is queried after round 3 of sub-session ℓ. At that point, the view of $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$—and hence its decision whether to output correct values $\{V_i\}$—depends on z only. It is only after the round-4 message of sub-session ℓ is sent that the view of $\mathcal{A}$ depends on h. We also continue to rely on the fact that a unique signature scheme is used, and hence the $\{\sigma_{z_i}\}$ are determined by the $\{\mathsf{com}_{z_i}\}$.) Thus, any difference in the output of $\mathcal{Z}$ based on the value of $z_i^T h_i$ would result in a difference in the output of A based on that information. Theorem 5 implies that the experiments are therefore indistinguishable by $\mathcal{Z}$.

Since $\mathcal{H}_8$ is identical to Ideal, this concludes the proof of Theorem 1 for a corrupted sender.

3.3.2 Corrupted Receiver

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest sender running π and the dummy adversary $\mathcal{A}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest sender, functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$, and Sim in the ideal world. Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{A}$.

Tokens. Sim simulates $\mathcal{A}$'s receipt of a token from the honest sender in the natural way, including generation of keys $(vk_{\mathsf{S}}, sk_{\mathsf{S}})$ for the signature scheme. It then waits to receive vk_{R} as well as a message $(\textsc{create}, \langle \mathsf{sid}, \mathsf{R}, \mathsf{S} \rangle, M_{\mathcal{A}})$ from $\mathcal{A}$ to the $\mathcal{F}_{\mathsf{wrap}}$ functionality, and records $M_{\mathcal{A}}$. For readability, we let $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ (resp., $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}} = M_{\mathcal{A}}$) denote the $\mathcal{F}_{\mathsf{wrap}}$ functionality corresponding to the sender's (resp., receiver's) token.

When $\mathcal{A}$ queries $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$, the simulator answers the query as in Figure 4, except that random functions are used in place of $\mathsf{PRF}_{k_a}(\cdot)$ and $\mathsf{PRF}_{k_B}(\cdot)$. A query $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ to $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ is said to be *valid* if $\mathsf{Vrfy}_{vk_{\mathsf{S}}}(\mathsf{ssid} \| i \| 0 \| \mathsf{com}_z, \sigma_z) = 1$ and $\mathsf{Open}(\mathsf{com}_z, z, r_z) = 1$. When a valid query $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ is made, the simulator records (ssid, i, z). If, at any point during the experiment, $\mathcal{A}$ makes a valid query $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ to $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ but σ_z was not previously generated by the simulator during sub-session ssid as a signature on com_z, the simulator outputs $\mathsf{abort1}$ and terminates. If $\mathcal{A}$ ever makes two valid queries $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ and $(\mathsf{ssid}, i, \mathsf{com}_z, z', r'_z, \sigma_z)$ with $z \neq z'$, Sim outputs $\mathsf{abort2}$ and terminates.

Simulation of the execution for sub-session ssid. The simulator Sim proceeds as follows:

1. In step 1, random functions are used in place of $\mathsf{PRF}_{k_a}(\cdot)$ and $\mathsf{PRF}_{k_B}(\cdot)$ to generate the $\{a_i\}$ and $\{B_i\}$. Sim otherwise runs this step honestly.

2. After receiving C and $\{(\mathsf{com}_{z_i}, \sigma_{a_i \| B_i})\}$ from $\mathcal{A}$, the simulator runs step 3 of the protocol honestly (including interacting with $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$), and aborts if dictated by the protocol.

3. After receiving $\{(h_i, \sigma_i)\}$ from $\mathcal{A}$, the simulator verifies the signatures just received and aborts if any of them is invalid. Otherwise, for each i it checks for a record of the form (ssid, i, z_i); note there can be at most one such record. If there is no such record, Sim aborts.

   Assuming it did not abort, Sim next computes $b_i := z_i^T h_i$ for all i and sends the message $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{b_i\}_{i=1}^m)$ to functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$. After receiving $\{x_i\}$ in return, Sim does the following for all i: choose keys v_i^0, v_i^1, set $\tilde{x}_i^{b_i} := \mathsf{Ext}(GB_ih_i + Ga_ib_i, v_i^{b_i}) \oplus x_i$, choose $\tilde{x}_i^{\bar{b}_i}$ uniformly, and send $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}$.

Let Real denote the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of π between an honest sender and $\mathcal{A}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the outputs of the honest sender.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest sender, functionality $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. In this experiment, we imagine an interaction between a simulator Sim' (playing the role of the receiver), the honest sender, and $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$ in the ideal world, but where Sim' is given the inputs $\{(x_i^0, x_i^1)\}$ of the sender at the outset of every sub-session. Sim' internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$, and running the protocol honestly with $\mathcal{A}$ using the sender's inputs. Valid queries are defined as above, and after a valid query $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ is made, Sim' records (ssid, i, z). When execution of a sub-session π is complete then, if the sub-session was not aborted, Sim' sets $b_i := 0$ for all i and sends $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R} \rangle, \mathsf{ssid}, \{b_i\}_{i=1}^m)$ to $\mathcal{F}_{\mathsf{unbounded\text{-}OT}}$.

It is immediate that the view of $\mathcal{Z}$ in $\mathcal{H}_1$ is distributed identically to its view in Real.

Experiment $\mathcal{H}_2$. This experiment is the same as the previous one, except that here Sim' uses random functions in place of $\mathsf{PRF}_{k_a}(\cdot)$ and $\mathsf{PRF}_{k_B}(\cdot)$. It is clear that if PRF is a pseudorandom function, then the view of $\mathcal{Z}$ here is computationally indistinguishable from its view in $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. This experiment is the same as the previous one except that if, at any point during the experiment, $\mathcal{A}$ makes a valid query $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ to $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ but σ_z was not previously generated by Sim' as a signature on $\mathsf{ssid}\|i\|0\|\mathsf{com}_z$, the simulator aborts. It is easy to see that strong unforgeability of the signature scheme means that the view of $\mathcal{Z}$ in this experiment is computationally indistinguishable from its view in $\mathcal{H}_2$.

Experiment $\mathcal{H}_4$. This experiment is the same as the previous one except that if $\mathcal{A}$ ever makes two valid queries $(\mathsf{ssid}, i, \mathsf{com}_z, z, r_z, \sigma_z)$ and $(\mathsf{ssid}, i, \mathsf{com}_z, z', r'_z, \sigma_z)$ to $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}}$ with $z \neq z'$ then the simulator aborts. It is easy to see that (computational) binding of the commitment scheme implies that the view of $\mathcal{Z}$ in this experiment is indistinguishable from its view in $\mathcal{H}_3$.

Experiment $\mathcal{H}_5$. This is identical to the previous experiment, except that upon receiving a message $\{(h_i, \sigma_i)\}$ from $\mathcal{A}$ the simulator proceeds as follows: It first verifies the signatures (as in the previous experiment), and aborts if they are invalid. Otherwise, for each i the simulator checks for a (unique) record of the form $(\mathsf{ssid}, i, \star)$. If there is no such record, then Sim' aborts. It is clear that strong unforgeability of the signature scheme means that the view of $\mathcal{Z}$ in this experiment is indistinguishable from its view in $\mathcal{H}_4$.

Experiment $\mathcal{H}_6$. This is just like the previous experiment, except that upon receiving a message $\{(h_i, \sigma_i)\}$ from $\mathcal{A}$ the simulator proceeds as follows for each i (assuming it has not yet aborted): find the unique record (ssid, i, z_i) and compute $b_i := z_i^T h_i$. Then choose keys v_i^0, v_i^1, set $\tilde{x}_i^{b_i} := \mathsf{Ext}(GB_i h_i + Ga_i b_i, v_i^{b_i}) \oplus x_i^{b_i}$, and choose $\tilde{x}_i^{\bar{b}_i}$ uniformly. We claim that the views of $\mathcal{Z}$ in experiments $\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically close.

To see this, we consider a sequence of intermediate experiments in which the computation of $\tilde{x}_i^0, \tilde{x}_i^1$ is changed for only a single index i in a single sub-session ssid; a standard hybrid argument then completes the proof. Consider the information that $\mathcal{A}$ has on the values a_i, B_i used in the sub-session in question. The statistically hiding commitment $\mathsf{com}_{a_i\|B_i}$ reveals nothing about a_i, B_i. From the protocol itself, $\mathcal{A}$ learns $\tilde{a}_i = Ca_i$, $\tilde{B}_i = CB_i$, and $V_i = a_i z_i^T + B_i$. As discussed in Section 3.1, the value $Z \overset{\text{def}}{=} GB_i h_i + Ga_i \bar{b}_i$ (where $b_i = z_i^T h_i$) is a uniform 2λ-bit value conditioned on this information, and hence $X \overset{\text{def}}{=} \mathsf{Ext}(Z, v_i^{\bar{b}_i})$ is statistically close to uniform even if we additionally condition on $v_i^{\bar{b}_i}$. But we still need to take into account information that $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ might leak to $\mathcal{A}$ about Z or X. We show next that $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ can leak no information about X, and at most $O(\log \lambda)$ bits of information about Z.

We first argue that the behavior of $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ is independent of X (given Z). This follows because all inputs provided to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ in all sub-sessions subsequent to ssid are independent of X given Z. This, in turn, is a consequence of the fact that all the inputs to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ except the final signature are chosen by the (honest) sender, but—because a unique signature scheme is used—that final signature is uniquely determined by the other inputs.

The behavior of $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ can depend on Z. However, because a unique signature scheme is used, the only dependence (besides the values $\tilde{a}_i, \tilde{B}_i$ already taken into account above) is in terms of if/when it ever aborts. Since at most polynomially many sub-sessions are run overall, we see that $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ can communicate only $O(\log \lambda)$ bits to $\mathcal{A}$ via such an abort.

We thus see that with overwhelming probability, the min-entropy of Z is at least 1.5λ, and hence the extracted value X is statistically close to uniform. This concludes the proof that the views of $\mathcal{Z}$ in $\mathcal{H}_6$ and $\mathcal{H}_5$ are statistically close.

Since $\mathcal{H}_6$ is identical to Ideal, this completes the proof of Theorem 1.

```
On input $(i, \mathsf{com}_z, z, r_z, \tau_z)$:
if $(\mathsf{Mac}_{s'}(i \| \mathsf{com}_z) = \tau_z$
    and $\mathsf{Open}(\mathsf{com}_z, z, r_z) = 1)$
        $V := a_i z^T + B_i$
        output $(V, w_i, r_{w_i})$
else output $\perp$
```

Figure 7: The Turing machine M_{S} to be embedded in the sender-created token $\mathcal{T}_{\mathsf{S}}$.

```
On input $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$:
if $\big(\mathsf{Mac}_s(i\|0\|\mathsf{com}_{a\|B}) = \tau_{a\|B}$
    and $\mathsf{Open}(\mathsf{com}_{a\|B}, a\|B, r_{a\|B}) = 1\big)$
        $\tilde{a} := Ca$
        $\tilde{B} := CB$
        $\tau_{\tilde{a}\|\tilde{B}} := \mathsf{Mac}_s(i\|1\|\tilde{a}\|\tilde{B})$
        output $(\tilde{a}, \tilde{B}, \tau_{\tilde{a}\|\tilde{B}})$
else output $\perp$
```

Figure 8: The Turing machine M_{R} to be embedded in the receiver-created token $\mathcal{T}_{\mathsf{R}}$.

3.4 Bounded Oblivious Transfer from Collision-Resistant Hash Functions

The requirement of unique signature schemes is a strong one, and in particular appears to require the use of public-key techniques. In this section we construct a protocol π' that can be based on symmetric-key primitives (MACs and collision-resistant hash functions) alone. Here, however, we are only able to realize the *bounded* OT functionality, in which an arbitrary number of OTs can be carried out, but must be done in parallel.

The main intuition for this protocol is as before: we modify the protocol of [20] so that it can be based on stateless tokens. The modifications are similar in spirit to those introduced in our prior protocol: at a high level, we prevent an attacker from querying a token more times than allowed by having tokens only respond to *authenticated* inputs. Here, however, we use message authentication codes (MACs) rather than (unique) signatures to provide this authentication. This introduces a potential covert channel between a malicious party and the token they create, since the other party (who does not hold the MAC key) cannot verify whether the MAC tag sent by the other party was honestly generated. We resolve this issue in part by having the parties commit to their MAC key in advance and sending the decommitment later. This allows the honest party to check, after the fact, whether the MAC tag was computed correctly. We also carefully order the steps of the protocol, ensuring in particular that the sender transmits its MAC tag before receiving C so that it cannot communicate information about C via an incorrectly computed tag. The necessity of doing so, however, is why we cannot support any sub-sessions after the first has been completed.

We now describe the protocol; see also Figure 9. As before, PRF denotes a pseudorandom function and $(\mathsf{SCom}, \mathsf{Open})$ is a non-interactive, statistically hiding commitment scheme. We also let Mac denote a deterministic message authentication code, and let $(\mathsf{Com}, \mathsf{Open})$ denote a non-interactive, statistically *binding* commitment scheme.

Token-exchange phase. Each party generates a single token and sends it to the other party. For $i = 1, \ldots, m$ (where m is the number of OTs to be supported), the sender chooses uniform $a_i \in \mathbb{F}_2^{4\lambda}$, $B_i \in \mathbb{F}_2^{4\lambda \times 4\lambda}$, $w_i \in \{0,1\}^\lambda$, $r_{w_i} \in \{0,1\}^*$; it also chooses a uniform MAC key s'; it then creates a token $\mathcal{T}_{\mathsf{S}}$ containing the code from Figure 7. The receiver chooses uniform $C \in \mathbb{F}_2^{2\lambda \times 4\lambda}$ and a uniform MAC key s; it then creates a token $\mathcal{T}_{\mathsf{R}}$ containing the code from Figure 8. The parties then exchange their tokens. (Each party also runs the initial rounds of the commitment schemes; we leave this implicit.)

Oblivious-transfer phase. Following the token-exchange phase, the parties execute m parallel

$\underline{\mathsf{S}}$		$\underline{\mathsf{R}}$
Exchange tokens and public keys: For $i \in [m]$: $a_i \leftarrow \mathbb{F}_2^{4\lambda}; B_i \leftarrow \mathbb{F}_2^{4\lambda \times 4\lambda}$ $w_i \leftarrow \{0,1\}^\lambda; r_{w_i} \leftarrow \{0,1\}^*$ $s' \leftarrow \{0,1\}^\lambda$ generate token $\mathcal{T}_\mathsf{S}$ as in Fig 7		$C \leftarrow \mathbb{F}_2^{2\lambda \times 4\lambda}; s \leftarrow \{0,1\}^\lambda$ generate token $\mathcal{T}_\mathsf{R}$ as in Fig 8
	$\xleftrightarrow{\mathcal{T}_\mathsf{S}; \mathcal{T}_\mathsf{R}}$	
Oblivious transfer: $\underline{(m, \{(x_i^0, x_i^1)\}_{i=1}^m)}$ For $i \in [m]$: $\mathsf{com}_{w_i} \leftarrow \mathsf{Com}(w_i; r_{w_i})$		$\underline{(m, \{b_i\}_{i=1}^m)}$
	$\xrightarrow{\{\mathsf{com}_{w_i}\}_{i=1}^m}$	
		$\mathsf{com}_s \leftarrow \mathsf{Com}(s; r_s)$ For $i \in [m]$: $h_i \leftarrow \mathbb{F}_2^{4\lambda} \setminus \{0\}$ $z_i \leftarrow \{z_i \in \mathbb{F}_2^{4\lambda} \mid z_i^T h_i = b_i\}$ $\mathsf{com}_{z_i} \leftarrow \mathsf{SCom}(z_i; r_{z_i})$
	$\xleftarrow{\mathsf{com}_s, \{\mathsf{com}_{z_i}\}_{i=1}^m}$	
For $i \in [m]$: $\tau_{z_i} := \mathsf{Mac}_{s'}(i \Vert \mathsf{com}_{z_i})$ $\mathsf{com}_{a_i \Vert B_i} \leftarrow \mathsf{SCom}(a_i \Vert B_i; r_{a_i \Vert B_i})$		
	$\xrightarrow{\{(\tau_{z_i}, \mathsf{com}_{a_i \Vert B_i})\}_{i=1}^m}$	
		For $i \in [m]$: $\tau_{a_i \Vert B_i} := \mathsf{Mac}_s(i \Vert 0 \Vert \mathsf{com}_{a_i \Vert B_i})$
	$\xleftarrow{C, \{\tau_{a_i \Vert B_i}\}_{i=1}^m}$	
For $i \in [m]$: Run $\mathcal{T}_\mathsf{R}(i, \mathsf{com}_{a_i \Vert B_i}, a_i, B_i, r_{a_i \Vert B_i}, \tau_{a_i \Vert B_i})$ to get $(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i \Vert \tilde{B}_i})$ If $\tilde{a}_i \neq C a_i$ or $\tilde{B}_i \neq C B_i$, abort		
	$\xrightarrow{\{(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i \Vert \tilde{B}_i})\}_{i=1}^m}$	
		For $i \in [m]$: If $\mathsf{Mac}_s(i \Vert 1 \Vert \tilde{a}_i \Vert \tilde{B}_i) \neq \tau_{\tilde{a}_i \Vert \tilde{B}_i}$, abort Run $\mathcal{T}_\mathsf{S}(i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})$ to get (V_i, w'_i, r'_{w_i}) If $\Big(\mathsf{Open}(\mathsf{com}_{w_i}, w'_i, r'_{w_i}) \neq 1$ or $C V_i \neq \tilde{a}_i z_i^T + \tilde{B}_i\Big)$ abort
	$\xleftarrow{(s, r_s), \{(h_i, w'_i)\}_{i=1}^m}$	
If $\mathsf{Open}(\mathsf{com}_s, s, r_s) \neq 1$, abort For $i \in [m]$: If $\Big(\mathsf{Mac}_s(i \Vert 1 \Vert \tilde{a}_i \Vert \tilde{B}_i) \neq \tau_{\tilde{a}_i \Vert \tilde{B}_i}$ or $w'_i \neq w_i\Big)$ abort $G := \mathsf{Comp}(C)$ For $i \in [m]$: choose keys v_i^0, v_i^1 $\tilde{x}_i^0 := \mathsf{Ext}(G B_i h_i, v_i^0) \oplus x_i^0$ $\tilde{x}_i^1 := \mathsf{Ext}(G B_i h_i + G a_i, v_i^1) \oplus x_i^1$		
	$\xrightarrow{\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}_{i=1}^m}$	
		$G := \mathsf{Comp}(C)$ For $i \in [m]$: output $\tilde{x}_i^{b_i} \oplus \mathsf{Ext}(G V_i h_i, v_i^{b_i})$

Figure 9: A bounded OT protocol π' from two stateless tokens.

OT instances. Let $\{(x_i^0, x_i^1)\}_{i=1}^m$ be the input of the sender S, and let $\{b_i\}_{i=1}^m$ denote the input of the receiver R. The protocol proceeds as follows:

Step 1: The sender commits to the $\{w_i\}$ using Com and randomness $\{r_{w_i}\}$, resulting in commitments $\{\mathsf{com}_{w_i}\}$. It sends $\{\mathsf{com}_{w_i}\}_{i=1}^m$ to R.

Step 2: The receiver commits to s using Com, resulting in commitment com_s and decommitment r_s. Then for $i \in [m]$ the receiver chooses uniform $h_i, z_i \in \mathbb{F}_2^{4\lambda}$ subject to $b_i = z_i^T h_i$. It commits to each z_i using SCom, resulting in commitment com_{z_i} and decommitment r_{z_i}. It sends com_s and $\{\mathsf{com}_{z_i}\}_{i=1}^m$ to S.

Step 3: S authenticates the $\{\mathsf{com}_{z_i}\}$ using its MAC key s', yielding MAC tags $\{\tau_{z_i}\}$ that it sends to R. It also commits to the $\{a_i, B_i\}$ using SCom, resulting in commitments $\{\mathsf{com}_{a_i\|B_i}\}$ and decommitments $\{r_{a_i\|B_i}\}$. It also sends $\{\mathsf{com}_{a_i\|B_i}\}_{i=1}^m$ to R.

Step 4: R authenticates the $\{\mathsf{com}_{a_i\|B_i}\}$ using its MAC key s, yielding MAC tags $\{\tau_{a_i\|B_i}\}$ that it sends to S. It also sends C.

Step 5: For $i \in [m]$, the sender runs the token $\mathcal{T}_{\mathsf{R}}$ on input $(i, \mathsf{com}_{a_i\|B_i}, a_i, B_i, r_{a_i\|B_i}, \tau_{a_i\|B_i})$, and obtains in return $(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i})$. It verifies that $\tilde{a}_i = Ca_i$ and $\tilde{B}_i = CB_i$ for all i, and aborts if not. Otherwise, it sends $\{(\tilde{a}_i, \tilde{B}_i, \tau_{\tilde{a}_i\|\tilde{B}_i})\}$ to R.

Step 6: R verifies the MAC tags just received, and aborts if any of them is invalid. Next, for all i it runs the token $\mathcal{T}_{\mathsf{S}}$ on input $(i, \mathsf{com}_{z_i}, z_i, r_{z_i}, \tau_{z_i})$, and obtains in return (V_i, w_i', r_{w_i}'). For all i, it then verifies that $\mathsf{Open}(\mathsf{com}_{w_i}, w_i', r_{w_i}') = 1$ and $CV_i = \tilde{a}_i z_i^T + \tilde{B}_i$, and aborts if not. If it has not aborted, then it sends (s, r_s) and $\{(h_i, w_i')\}_{i=1}^m$ to S.

Step 7: S verifies that the $\{w_i'\}$ just received are equal to the $\{w_i\}$ and that $\mathsf{Open}(\mathsf{com}_s, s, r_s) = 1$, and aborts if not. It then uses s to verify the MAC tags $\{\tau_{\tilde{a}_i\|\tilde{B}_i}\}$, and aborts if any is found to be invalid. Assuming it has not aborted, it computes $G := \mathsf{Comp}(C)$ and then for all i chooses uniform extractor keys v_i^0, v_i^1 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 $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}$ to R.

Step 8 (output determination): The receiver computes $G := \mathsf{Comp}(C)$ and then, for $i \in [m]$, outputs $x_i^{b_i} := \mathsf{Ext}(\tilde{x}_i^{b_i}, v_i^{b_i}) - GV_ih_i$.

We sketch the intuition for why the protocol is secure, assuming the reader is already familiar with the proof from the previous section.

Corrupt sender. During an execution of the protocol, a malicious sender S^* learns C and h_i but we claim that it learns at most one bit of information about z_i. Indeed, the only way it can potentially learn information about z_i is from the values $\{w_i'\}$ output by the (potentially malicious) token $\mathcal{T}_{\mathsf{S}^*}$. But these values must be equal to the values to which S^* commits in the first round of the protocol, or else R aborts. Thus, $\mathcal{T}_{\mathsf{S}^*}$ is limited to causing R to abort, which in turn communicates just a single bit of information to S^*.

In the formal proof we also need to show how to extract the effective inputs of S^* from an execution. This is done as in the previous section, by exploiting the fact that S^* must query $\mathcal{T}_{\mathsf{R}}$ on a single valid input or else R will abort the protocol.

Corrupt receiver. Here, we need to bound the information that a malicious receiver R^* learns about a_i, B_i (beyond $\tilde{a}_i, \tilde{B}_i$). The only potential information R^* can learn is from the output of the (potentially malicious) token $\mathcal{T}_{\mathsf{R}^*}$—and, indeed, the token could potentially embed a_i, B_i themselves into the MAC tag $\tau_{\tilde{a}_i\|\tilde{B}_i}$ that it outputs. However, as long as the sender does not abort the protocol, the MAC tag output by $\mathcal{T}_{\mathsf{R}^*}$ is a deterministic function of the key s (to which R^* committed in the second round) and the commitment $\mathsf{com}_{\tilde{a}_i\|\tilde{B}_i}$. We thus see that—if the protocol is not aborted—R^* learns at most one bit of information about the $\{a_i, B_i\}$. (And if the protocol is aborted, then R^* learns nothing about the sender's inputs.) Since we apply a strong extractor to the values GB_ih_i and $GB_ih_i + Ga_i$, we conclude that R^* learns no information about one of the sender's inputs.

In the formal proof we also need to show how to extract the effective inputs of R^*. This is done as in the previous section, by exploiting the fact that R^* must query $\mathcal{T}_{\mathsf{S}}$ on a single valid input or else S will abort the protocol. (We highlight here that the $\{w_i\}$ values play the role of "MAC tags," in the sense that they prove the token was queried on some valid inputs.)

3.5 Proof of Security for Bounded Oblivious Transfer Protocol

Theorem 2 *If* Com *is statistically binding,* SCom *is statistically hiding,* Mac *is a secure message authentication code, and* Ext *is a* $(1.5\lambda, \mathsf{negl}(\lambda))$*-strong extractor, then the protocol of Figure 9 securely realizes* $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$ *in the* $\mathcal{F}_{\mathsf{wrap}}$*-hybrid model.*

To prove the theorem, we construct a straight-line simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish between (1) an execution involving an honest party and a corrupted party (that without loss of generality we may take as the dummy adversary who simply forwards messages to/from $\mathcal{Z}$) running protocol π' in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving the same honest party, functionality $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, and Sim in the ideal world. We consider the cases of a corrupted sender and a corrupted receiver separately.

3.5.1 Corrupted Sender

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest receiver running π' and the dummy adversary $\mathcal{A}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest receiver, functionality $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, and Sim in the ideal world. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{A}$.

Tokens. Sim chooses uniform C and s, and then simulates $\mathcal{A}$'s receipt of a token from the honest receiver in the natural way. It then waits to receive a message $(\textsc{create}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, M_{\mathcal{A}})$ from $\mathcal{A}$ to the $\mathcal{F}_{\mathsf{wrap}}$ functionality, and records $M_{\mathcal{A}}$. For readability, we let $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ (resp., $\mathcal{F}^{\mathsf{S}}_{\mathsf{wrap}} = M_{\mathcal{A}}$) denote the $\mathcal{F}_{\mathsf{wrap}}$ functionality corresponding to the receiver's (resp., sender's) token.

Whenever $\mathcal{A}$ queries $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$, the simulator Sim answers the query as in Figure 8. A query $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ is said to be *valid* if $\mathsf{Mac}_s(i\|0\|\mathsf{com}_{a\|B}) = \tau_{a\|B}$ and furthermore $\mathsf{Open}(\mathsf{com}_{a\|B}, a\|B, r_{a\|B}) = 1$. When a valid query is answered with $(\tilde{a}, \tilde{B}, \tau_{\tilde{a}\|\tilde{B}})$, the simulator records (i, a, B). If $\mathcal{A}$ ever makes a valid query $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ to $\mathcal{F}^{\mathsf{R}}_{\mathsf{wrap}}$ but $\tau_{a\|B}$ was not previously generated by the simulator as a MAC tag on $\mathsf{com}_{a\|B}$, or if $\mathcal{A}$ ever makes two valid queries $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ and $(i, \mathsf{com}_{a\|B}, a', B', r'_{a\|B}, \tau_{a\|B})$ with $(a, B) \neq (a', B')$, the simulator aborts.

Simulation of the protocol. The simulator Sim runs the entire protocol exactly as an honest receiver would using input $b_i = 0$ for all i. Immediately before sending its message in step 6 of the protocol (assuming it has not already aborted), Sim searches for a record of the form (i, a_i, B_i) for each i; if there is not a unique such record, or if there is such a record but $Ca_i \neq \tilde{a}_i$ or $CB_i \neq \tilde{B}_i$, then Sim aborts. Otherwise, it continues running the protocol to the end. It then computes $x_i^0 := \tilde{x}_i^0 \oplus \mathsf{Ext}(GB_ih_i, v_i^0)$ and $x_i^1 := \tilde{x}_i^1 \oplus \mathsf{Ext}(GB_ih_i + Ga_i, v_i^1)$ for all i, and sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \{(x_i^0, x_i^1)\}_{i=1}^m)$ to functionality $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$.

Let Real denote the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of π' between an honest receiver and $\mathcal{A}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the outputs of the honest receiver.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest receiver, $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. In this experiment, we imagine an interaction between a simulator Sim' (playing the role of the sender), the honest receiver and $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$ in the ideal world, but where Sim' is given the inputs $\{b_i\}_{i=1}^m$ of the receiver. Sim' internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. It also simulates an execution of protocol π' with $\mathcal{A}$, by playing the role of the receiver itself using the inputs $\{b_i\}_{i=1}^m$. As part of this simulation, Sim' emulates $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ and interacts with $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$. A query to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ is define to be *valid* as before. When a valid query $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ is made, the simulator records (i, a, B).

When execution of π' is complete then, if it was not aborted, Sim' obtains outputs $\{x_i^{b_i}\}_{i=1}^m$. It sets $x_i^{\bar{b}_i} := 0^\lambda$ for $i \in [m]$ and sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \{(x_i^0, x_i^1)\}_{i=1}^m)$ to functionality $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$.

It is immediate that the view of $\mathcal{Z}$ in $\mathcal{H}_1$ is distributed identically to its view in Real.

Experiment $\mathcal{H}_2$. This experiment is as before except that now if $\mathcal{A}$ ever makes a valid query $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ but $\tau_{a\|B}$ was not previously generated by the simulator as a MAC tag on $\mathsf{com}_{a\|B}$, or if the adversary $\mathcal{A}$ ever makes two valid queries $(i, \mathsf{com}_{a\|B}, a, B, r_{a\|B}, \tau_{a\|B})$ and $(i, \mathsf{com}_{a\|B}, a', B', r'_{a\|B}, \tau_{a\|B})$ with $(a, B) \neq (a', B')$, the simulator aborts. Security of the MAC and binding of the commitment scheme imply that the view of $\mathcal{Z}$ here is indistinguishable from its view in $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. This is identical to the previous experiment, except that immediately before sending its message in step 6 of the protocol (assuming it has not already aborted), Sim' searches for a record of the form (i, a_i, B_i) for each i; if there is not a unique such record, or if there is such a record but $Ca_i \neq \tilde{a}_i$ or $CB_i \neq \tilde{B}_i$, then Sim aborts. Security of the MAC means that the view of $\mathcal{Z}$ here is indistinguishable from its view in $\mathcal{H}_2$.

Experiment $\mathcal{H}_4$. As in the previous experiment, in step 6, Sim' finds the unique record of the form (i, a_i, B_i) for each i Now, however, Sim' aborts if $V_i \neq a_i z_i^T + B_i$; otherwise, it proceeds as before. We claim that the view of $\mathcal{Z}$ in $\mathcal{H}_4$ is statistically close to its view in $\mathcal{H}_3$. To see this, recall that the simulator aborts in $\mathcal{H}_3$ and $\mathcal{H}_4$ if $\tilde{a}_i \neq Ca_i$ or $\tilde{B}_i \neq CB_i$; in $\mathcal{H}_3$, the simulator aborts only if $CV_i \neq \tilde{a}_i z_i^T + \tilde{B}_i$. Thus, the only difference between $\mathcal{H}_3$ and $\mathcal{H}_4$ occurs if there is ever a sub-session and an index i for which $CV_i = \tilde{a}_i z_i^T + \tilde{B}_i = C \cdot \left(a_i z_i^T + B_i\right)$ but $V_i \neq a_i z_i^T + B_i$. As in the analysis of experiment $\mathcal{H}_6$ in Section 3.3.1, one can show that this occurs with negligible probability. (We remark that rather than relying on uniqueness of the signature scheme, here we use the fact that the τ_{z_i} are sent by $\mathcal{A}$ before it learns C.)

Experiment $\mathcal{H}_5$. In this experiment, Sim' uses the values a_i, B_i that it derived[1] in step 6 to compute $x_i^0 := \tilde{x}_i^0 \oplus \mathsf{Ext}(GB_ih_i, v_i^0)$ and $x_i^1 := \tilde{x}_i^1 \oplus \mathsf{Ext}(Ga_i + GB_ih_i, v_i^1)$ in step 8. The rest of its execution is as before.

We claim that the views of $\mathcal{Z}$ in $\mathcal{H}_4$ and $\mathcal{H}_5$ are identical. The change in the computation of $x_i^{\bar{b}_i}$ does not affect the view of $\mathcal{Z}$ at all (since the honest party's input is b_i). We consider next the change in the computation of $x_i^{b_i}$. In both $\mathcal{H}_4$ and $\mathcal{H}_5$, the simulator aborts unless $V_i = a_iz_i^T + B_i$. In step 8 of $\mathcal{H}_4$, the simulator computes $x_i^{b_i} := \tilde{x}_i^{b_i} \oplus \mathsf{Ext}(GV_ih_i, v_i^{b_i})$. But then in $\mathcal{H}_5$ we have

$$GV_ih_i = G \cdot (a_iz_i^T + B_i)h_i = Ga_ib_i + GB_ih_i,$$

and so the result computed for $x_i^{b_i}$ is the same.

Experiment $\mathcal{H}_6$. This is identical to the previous experiment, except that now Sim' chooses $z_i \leftarrow \{z \in \mathbb{F}_2^{4\lambda} \mid z^Th_i = 0\}$ for all i. We claim that the distributions on the view of $\mathcal{Z}$ in $\mathcal{H}_5$ and $\mathcal{H}_6$ are statistically close. The argument here is much simpler than in the previous section, since only a single session is executed. It suffices to note that the view of $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ is independent of the $\{h_i\}$, and the view of $\mathcal{A}$ is independent of the $\{z_i\}$ except possibly for one bit of information that $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ can transmit by aborting. (Here we use the fact that $\mathcal{A}$ commits to the $\{w_i\}$ in advance.) Viewing $z_i^Th_i$ as a strong extractor implies that b_i remains statistically close to uniform.

Since $\mathcal{H}_6$ is identical to Ideal, this completes the proof of Theorem 2 for a corrupted sender.

3.5.2 Corrupted Receiver

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest sender running π' and the dummy adversary $\mathcal{A}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest sender, $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, and Sim in the ideal world. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{A}$.

Tokens. Sim chooses $\{a_i, B_i, w_i, r_{w_i}\}$ and s' as an honest sender would, and then simulates $\mathcal{A}$'s receipt of the sender's token in the natural way. It then waits to receive $(\text{CREATE}, \langle \mathsf{sid}, \mathsf{R}, \mathsf{S}\rangle, M_{\mathcal{A}})$ from $\mathcal{A}$ to the $\mathcal{F}_{\mathsf{wrap}}$ functionality, and records $M_{\mathcal{A}}$. For readability, we let $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ (resp., $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}} = M_{\mathcal{A}}$) denote the $\mathcal{F}_{\mathsf{wrap}}$ functionality corresponding to the sender's (resp., receiver's) token.

When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$, the simulator answers the query as in Figure 7. A query $(i, \mathsf{com}_z, z, r_z, \tau_z)$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ is said to be *valid* if $\mathsf{Mac}_{s'}(i\|\mathsf{com}_z) = \tau_z$ and $\mathsf{Open}(\mathsf{com}_z, z, r_z) = 1$. When a valid query $(i, \mathsf{com}_z, z, r_z, \tau_z)$ is made, the simulator records (i, z). If, at any point during the experiment, $\mathcal{A}$ makes a valid query $(i, \mathsf{com}_z, z, r_z, \tau_z)$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ but τ_z was not previously generated by the simulator as a MAC tag on com_z, or if $\mathcal{A}$ ever makes two valid queries $(i, \mathsf{com}_z, z, r_z, \tau_z)$ and $(i, \mathsf{com}_z, z', r'_z, \tau_z)$ with $z \neq z'$, the simulator aborts.

Simulation of the protocol. The simulator runs the protocol as as honest sender would until step 7. At that point (assuming it has not yet aborted), for each i it finds a record of the form (i, z_i); note there can be at most one such record. If there is no such record, Sim aborts. Otherwise, it computes $b_i := z_i^Th_i$ for all i and sends $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \{b_i\}_{i=1}^m)$ to functionality $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$. After receiving $\{x_i\}$ in return, Sim does the following for all i: choose keys v_i^0, v_i^1, set $\tilde{x}_i^{b_i} := \mathsf{Ext}(GB_ih_i + Ga_ib_i, v_i^{b_i}) \oplus x_i$, choose $\tilde{x}_i^{\bar{b}_i}$ uniformly, and send $\{(v_i^0, v_i^1, \tilde{x}_i^0, \tilde{x}_i^1)\}$.

[1] Note that by step 8, there may be multiple valid queries to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ using different values of $\mathsf{com}_{a_i\|B_i}$. But in step 6, before the MAC key s is sent to $\mathcal{A}$, that cannot occur except with negligible probability.

Let Real denote the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of π between an honest sender and $\mathcal{A}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the outputs of the honest sender.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest sender, $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. In this experiment, we imagine an interaction between a simulator Sim' (playing the role of the receiver), the honest sender, and $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$, in the ideal world, but where Sim' is given the inputs $\{(x_i^0, x_i^1)\}$ of the sender. Sim' internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$, and running the protocol honestly with $\mathcal{A}$ using the sender's inputs. Valid queries are defined as above, and after a valid query $(i, \mathsf{com}_z, z, r_z, \tau_z)$ is made, Sim' records (i, z). When execution of the protocol is complete then, if it did not abort, Sim' sets $b_i := 0$ for all i and sends $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{S}, \mathsf{R}\rangle, \{b_i\}_{i=1}^m)$ to $\mathcal{F}_{\mathsf{bounded\text{-}OT}}$.

It is immediate that the view of $\mathcal{Z}$ in $\mathcal{H}_1$ is distributed identically to its view in Real.

Experiment $\mathcal{H}_2$. This is identical to the previous experiment, except that if $\mathcal{A}$ ever makes a valid query $(i, \mathsf{com}_z, z, r_z, \tau_z)$ to $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{S}}$ but τ_z was not previously generated by the simulator as a MAC tag on com_z, or if $\mathcal{A}$ ever makes two valid queries $(i, \mathsf{com}_z, z, r_z, \tau_z)$ and $(i, \mathsf{com}_z, z', r_z', \tau_z)$ with $z \neq z'$, the simulator aborts. It follows from security of the MAC and binding of the commitment scheme that the view of $\mathcal{Z}$ in this experiment is indistinguishable from its view in $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. This is just like the previous experiment, except that in step 7, assuming it does not abort, the simulator finds a record of the form (i, z_i) for all i. If there is no such record for some i, the simulator aborts. Computational hiding of the commitment scheme Com, along with the fact that the $\{w_i\}$ are uniform λ-bit values, implies that the view of $\mathcal{Z}$ in experiment $\mathcal{H}_3$ is indistinguishable from its view in $\mathcal{H}_2$.

Experiment $\mathcal{H}_4$. We modify the simulator as follows. For each i, after finding a record of the form (i, z_i) in step 7, Sim' computes $b_i := z_i^T h_i$. Next, it chooses keys v_i^0, v_i^1, sets $\tilde{x}_i^{b_i} := \mathsf{Ext}(GB_ih_i + Ga_ib_i, v_i^{b_i}) \oplus x_i^{b_i}$, and chooses $\tilde{x}_i^{\bar{b}_i}$ uniformly. We claim that the views of $\mathcal{Z}$ in experiments $\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically close.

To see this, consider the information that $\mathcal{A}$ has about a_i, B_i for some particular index i. The statistically hiding commitment $\mathsf{com}_{a_i \| B_i}$ reveals nothing about a_i, B_i. From the protocol itself, $\mathcal{A}$ learns $\tilde{a}_i = Ca_i$, $\tilde{B}_i = CB_i$, and $V_i = a_i z_i^T + B_i$. As discussed in Section 3.1, the value $Z = GB_ih_i + Ga_i\bar{b}_i$ (where $b_i = z_i^T h_i$) is a uniform 2λ-bit value conditioned on this information. Moreover, $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ can leak at most one bit of information—depending on whether or not S aborts—to $\mathcal{A}$ about a_i, B_i (beyond what is already implied by $\tilde{a}_i, \tilde{B}_i$), and hence about Z. This is because if S does not abort, then $\tau_{\tilde{a}_i \| \tilde{B}_i}$ is a deterministic function of com_s and $\tilde{a}_i, \tilde{B}_i$. (If S aborts then $\mathcal{F}_{\mathsf{wrap}}^{\mathsf{R}}$ may have leaked information about a_i, B_i to $\mathcal{A}$, but in that case $\mathcal{A}$ learns nothing about x_i^0, x_i^1.) We thus see that with overwhelming probability, the min-entropy of Z is at least 1.5λ, and hence the extracted value is statistically close to uniform. This concludes the proof that the views of $\mathcal{Z}$ in $\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically close.

Since $\mathcal{H}_4$ is identical to Ideal, this completes the proof of Theorem 2.

4 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 (even a single) universally composable OT using *one* stateless token is impossible using black-box techniques alone. Here, "black-box" refers to the simulator's access to the code $\mathcal{M}$ encapsulated in the token. (We formalize this below.) Note that the token model as defined by Katz [39] is inherently *non*black-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 proofs of security for our protocols in Section 3—as well as in most [47, 26, 16, 28, 29, 20] (though not all [13]) prior work—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 the code itself.

Formally, we consider simulators of the form $\mathsf{Sim} = (\mathsf{Sim}_{code}, \mathsf{Sim}_{bb})$, where Sim_{bb} is the main simulator routine and Sim_{code} merely gets the code $\mathcal{M}$ that the adversary submits to $\mathcal{F}_{\mathsf{wrap}}$ and provides Sim_{bb} with (black-box) oracle access to $\mathcal{M}$. (In addition, as required in the setting of universal composability, Sim_{bb} must be straight-line.) Inspired by the works of Canetti et al. [9, 10] we show that, restricting to such simulators, constructions of OT from one stateless token are impossible. Intuitively, for any such protocol, Sim_{bb} must be able to extract the input of a corrupted token-creating party after interacting with $\mathcal{M}$ in a black-box fashion. The real-world adversary can then use Sim_{bb} to extract the input of the honest token-creating party by running Sim_{bb} and answering its oracle queries by *querying the token* itself; for *stateless* tokens, querying the token (in the real world) is equivalent to black-box access to $\mathcal{M}$ (in the ideal world).

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 black-box simulator as defined above.*

Proof (Sketch) Let π be a protocol between a sender S and a receiver R, in which a single token is sent from the sender to the receiver. (The other case is handled analogously.) Consider the following environment $\mathcal{Z}'$ and dummy adversary $\mathcal{A}'$ corrupting the sender: $\mathcal{Z}'$ chooses uniform x_0, x_1, and b, and instructs the sender to run π honestly on input (x_0, x_1). The receiver is given input b. Once the honest receiver outputs x, then $\mathcal{Z}'$ outputs 1 iff $x = x_b$. Note that correctness implies that $\mathcal{Z}'$ outputs 1 with overwhelming probability.

Suppose π securely realizes $\mathcal{F}_{\mathsf{OT}}$, where security is proved via a black-box simulator $\mathsf{Sim} = (\mathsf{Sim}_{code}, \mathsf{Sim}_{bb})$ as previously described. In the course of the proof, Sim_{bb} *plays the role of a receiver* while interacting with $\mathcal{A}'$, and Sim_{code} provides Sim_{bb} with black-box access to the code $\mathcal{A}'$ submits to $\mathcal{F}_{\mathsf{wrap}}$. At some point during its execution, Sim_{bb} must send some inputs $(\tilde{x}_0, \tilde{x}_1)$ to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$. It is not hard to see that we must have $(\tilde{x}_0, \tilde{x}_1) = (x_0, x_1)$ with all but negligible probability.

We now consider a different environment $\mathcal{Z}$ and an adversary $\mathcal{A}$ corrupting the receiver. $\mathcal{Z}$ chooses uniform x_0, x_1, b and provides (x_0, x_1) as input to the honest sender. Adversary $\mathcal{A}$ works as follows (note that $\mathcal{A}$ receives a token from the honest sender as specified by π):

> Run Sim_{bb}, relaying messages from the honest sender to this internal copy of Sim_{bb}. Whenever Sim_{bb} makes a query to Sim_{code} to run $\mathcal{M}$ (the code created by the honest sender), $\mathcal{A}$ runs the token on the same query and gives the response to Sim_{bb}. At some point, Sim_{bb} sends $(\tilde{x}_0, \tilde{x}_1)$ to $\mathcal{F}_{\mathsf{OT}}$, at which point $\mathcal{A}$ outputs $(\tilde{x}_0, \tilde{x}_1)$ and halts.

$\mathcal{Z}$ outputs 1 iff $\mathcal{A}$ outputs (x_0, x_1).

The key point is that *since the token is stateless*, there is no difference between Sim_{code} running the code $\mathcal{M}$, and $\mathcal{A}$ querying its token. Thus, $\mathcal{A}$ provides a perfect simulation for Sim_{bb}, and so $(\tilde{x}_0, \tilde{x}_1) = (x_0, x_1)$ 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 security of π. ■

We remark that the proof can be easily extended to show that *at least one token must be sent in each direction*; that is, it does not help to use multiple tokens if they are all sent from the sender to the receiver.

5 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 possible using *non*black-box techniques. Here, we find it simpler to construct a protocol for universally composable coin tossing rather than OT. (The coin-tossing functionality $\mathcal{F}_{\mathsf{coin}}$ is defined in the natural way; see Figure 10.) UC coin tossing suffices for general UC computation under a variety of cryptographic assumptions [11, 43].

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

$\mathcal{F}_{\mathsf{coin}}$ interacts with two parties, A and 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 $(\textsc{toss}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle)$ from party $P \in \{\mathsf{A}, \mathsf{B}\}$, set $b_P := \mathsf{true}$, and send $(\textsc{toss}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, P)$ to the adversary. Ignore further $(\textsc{toss}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle)$ inputs from P.

Upon receiving a message $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, P)$ 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, if $\mathsf{coins} = \perp$, choose uniform $\mathsf{coins} \in \{0,1\}^{\lambda}$. Then return $(\textsc{coin}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, \mathsf{coins})$ to party P. Ignore further $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, P)$ messages for P.

Figure 10: The $\mathcal{F}_{\mathsf{coin}}$ functionality.

At a high level, our protocol follows the general structure of Blum's coin-tossing protocol [5]. This protocol consists of three moves. In the first move (B1), A generates a commitment com_x to a uniform value x, and B replies with a uniform value y (B2). In the last move (B3), A 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 A is malicious the simulator needs to extract the value x contained in com_x (extractability), whereas when B is corrupted the simulator needs to open the commitment in an arbitrary way (equivocation). We achieve both goals by having B send a single stateless token $\mathcal{T}_{\mathsf{B}}$ to A. 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 B that A 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 3, the token only works on authenticated inputs. That is, whenever A wants to query the token on some input, she needs to first ask B to compute a MAC on that input. For extractability, we have A query the token on input $(\mathsf{com}_x, x, r_x, \tau_x)$, where r_x is the decommitment for com_x 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 B that A knows the decommitment. As in the previous OT protocol, authentication of the inputs guarantees that A makes exactly one valid query to the token. We can then 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) A commits to x by executing $\mathsf{com}_x \leftarrow \mathsf{SCom}(x; r_x)$. She sends the commitment com_x to B, who in turn computes a tag τ_x on com_x and sends τ_x to A. 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. To allow for equivocation, we further modify the protocol as follows. A sends com_x and a dummy commitment com_M and gets a tag τ_x on $\mathsf{com}_x \| \mathsf{com}_M$ from B. The token is modified so it takes two types of inputs: on input $(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ it outputs a random value e, and on input $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ it outputs a random value t. In step (B3), instead of sending the decommitment, A now sends x together with a witness-indistinguishable (WI) proof that either x is a valid decommitment of com_x, or that com_M contains code that outputs the actual output e of the token $\mathcal{T}_{\mathsf{B}}$. We use the nonblack-box techniques of Barak [3] to carry out this WI proof.

Due to the binding property of com_M, and because e is unpredictable, a real-world A cannot commit to code that outputs e. The simulator, however, can take advantage of the fact that it obtains the code of the token generated by B. As in [3], the simulator's ability to predict the output of the token beforehand is 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.

5.1 Formal Description of the Protocol

The protocol ψ between A and B consists of an initial token-exchange phase, followed by a coin-tossing phase for generating a λ-bit string. We now describe ψ formally; see also Figure 11.

Token-exchange phase. B chooses uniform $s, e, t, r_t \in \{0,1\}^\lambda$, generates a token $\mathcal{T}_{\mathsf{B}}$ encapsulating the code described in Figure 12, and sends the token to A.

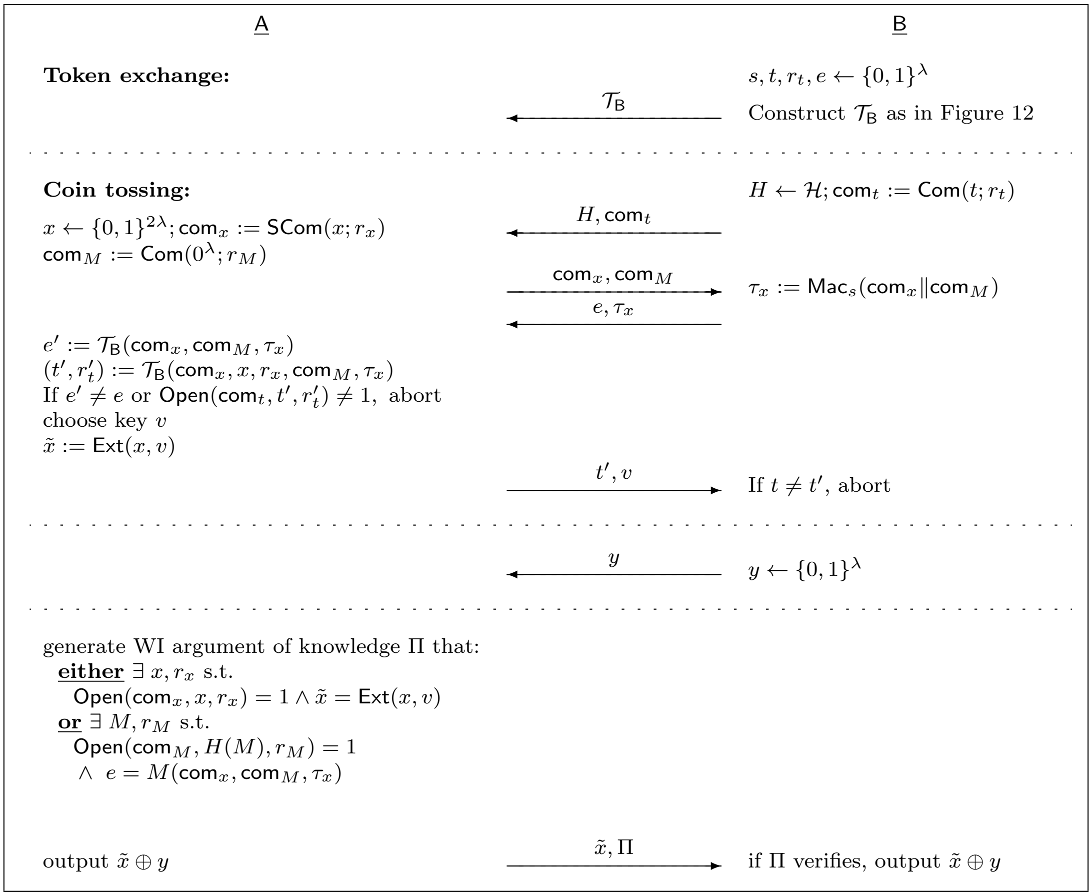
Coin-tossing phase. Let Com denote a statistically binding commitment scheme, SCom denote a statistically hiding commitment scheme, and Mac denote a secure MAC. In this phase, Alice and Bob proceed as follows.

Step 1: B chooses a collision-resistant hash function $H \leftarrow \mathcal{H}$, and also commits to t using Com, resulting in commitment com_t and decommitment r_t. It sends H, com_t to A.

Step 2: A chooses uniform $x \in \{0,1\}^{2\lambda}$ and commits to x by running $\mathsf{com}_x := \mathsf{SCom}(x; r_x)$. It commits to 0^λ by running $\mathsf{com}_M := \mathsf{Com}(0^\lambda; r_M)$, and sends com_x and com_M to B.

Step 3: B computes $\tau_x := \mathsf{Mac}_s(\mathsf{com}_x \| \mathsf{com}_M)$ and sends τ_x and e to A.

Step 4: A runs $\mathcal{T}_{\mathsf{B}}(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ to obtain response e'. Next, it runs $\mathcal{T}_{\mathsf{B}}(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ to obtain response (t', r'_t). A checks if $e' = e$ and $\mathsf{Open}(\mathsf{com}_t, t', r'_t) = 1$ and aborts if these do not hold. Otherwise, it chooses a uniform extractor key v and sends (t', v) to B.



A	B

Token exchange: $s, t, r_t, e \leftarrow \{0,1\}^\lambda$

$\xleftarrow{\mathcal{T}_\mathsf{B}}$ Construct $\mathcal{T}_\mathsf{B}$ as in Figure 12

Coin tossing: $H \leftarrow \mathcal{H}; \mathsf{com}_t := \mathsf{Com}(t; r_t)$

$\xleftarrow{H, \mathsf{com}_t}$

$x \leftarrow \{0,1\}^{2\lambda}; \mathsf{com}_x := \mathsf{SCom}(x; r_x)$

$\mathsf{com}_M := \mathsf{Com}(0^\lambda; r_M)$

$\xrightarrow{\mathsf{com}_x, \mathsf{com}_M}$ $\tau_x := \mathsf{Mac}_s(\mathsf{com}_x \| \mathsf{com}_M)$

$\xleftarrow{e, \tau_x}$

$e' := \mathcal{T}_\mathsf{B}(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$

$(t', r_t') := \mathcal{T}_\mathsf{B}(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$

If $e' \neq e$ or $\mathsf{Open}(\mathsf{com}_t, t', r_t') \neq 1,$ abort

choose key v

$\tilde{x} := \mathsf{Ext}(x, v)$

$\xrightarrow{t', v}$ If $t \neq t'$, abort

$\xleftarrow{y}$ $y \leftarrow \{0,1\}^\lambda$

generate WI argument of knowledge Π that:

either $\exists\, x, r_x$ s.t.

$\mathsf{Open}(\mathsf{com}_x, x, r_x) = 1 \wedge \tilde{x} = \mathsf{Ext}(x, v)$

or $\exists\, M, r_M$ s.t.

$\mathsf{Open}(\mathsf{com}_M, H(M), r_M) = 1$

$\wedge\ e = M(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$

output $\tilde{x} \oplus y$ $\xrightarrow{\tilde{x}, \Pi}$ if Π verifies, output $\tilde{x} \oplus y$

Figure 11: A coin-tossing protocol ψ from a single stateless token.

Step 5: B checks that $t = t'$, and aborts if not. Otherwise it chooses uniform $y \in \{0,1\}^\lambda$ and sends y to A.

Step 6: A sends $\tilde{x} := \mathsf{Ext}(x, v)$ and gives an interactive WI argument of knowledge that

- **either** there exist x, r_x such that $\mathsf{Open}(\mathsf{com}_x, x, r_x) = 1$ and $\tilde{x} = \mathsf{Ext}(x, v)$.
- **or** there exist M, r_M such that $\mathsf{Open}(\mathsf{com}_M, H(M), r_M) = 1$ and, treating M as the description of a Turing machine, $M(\mathsf{com}_x, \mathsf{com}_M, \tau_x)$ outputs e in time at most $p(\lambda)$, where p is the running time defined by $\mathcal{F}_\mathsf{wrap}$.

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

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

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

Figure 12: The Turing machine M embedded in the sender-created token $\mathcal{T}_\mathsf{B}$.

We sketch the intuition for the proof of security.

Corrupted A. Since e is uniform, A cannot commit to M that outputs e (except with negligible probability). Thus, we must have $\tilde{x} = \mathsf{Ext}(x, v)$. The simulator extracts x (and hence learns $\tilde{x}$) by finding the unique valid query of the form $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ that A makes to $\mathcal{T}_{\mathsf{B}}$. Having done so, the simulator can then set $y := \tilde{x} \oplus \mathsf{coins}$, where coins is the value it received from $\mathcal{F}_{\mathsf{coin}}$.

Corrupted B. The simulator obtains the code encapsulated in $\mathcal{T}_{\mathsf{B}}$, and commits to that code in com_M. This allows the simulator to equivocate $\tilde{x}$ to any desired value.

5.2 Proof of Security

Theorem 4 *If Com is statistically binding, MAC is a secure message authentication code, $\mathcal{H}$ is collision resistant, Ext is a $(1.5\lambda, \mathsf{negl}(\lambda))$-strong extractor, and the proof system is a WI argument of knowledge, then ψ securely realizes $\mathcal{F}_{\mathsf{coin}}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid model.*

To prove the theorem, we construct a straight-line simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish between (1) an execution involving an honest party and a corrupted party (that without loss of generality we may take as the dummy adversary who simply forwards messages to/from $\mathcal{Z}$) running protocol ψ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving the same honest party, functionality $\mathcal{F}_{\mathsf{coin}}$, and Sim in the ideal world. We consider the cases of a corrupted A and a corrupted B separately.

5.2.1 Corrupted A

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest B running ψ and the dummy adversary $\mathcal{A}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest B, functionality $\mathcal{F}_{\mathsf{coin}}$, and Sim in the ideal world. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{A}$.

Token. Sim chooses uniform s, e, t, r_t and simulates $\mathcal{A}$'s receipt of a token from the honest B in the natural way. We let $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ denote this token. Whenever $\mathcal{A}$ queries $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$, the simulator Sim answers the query as in Figure 12. A query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ to $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ is said to be *valid* if $\mathsf{Mac}_s(\mathsf{com}_x \| \mathsf{com}_M) = \tau_x$ and $\mathsf{Open}(\mathsf{com}_x, x, r_x) = 1$. When $\mathcal{A}$ makes a valid query, Sim records x.

If $\mathcal{A}$ ever makes a valid query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ to $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ but τ_x was not previously generated by the simulator as a MAC tag on $\mathsf{com}_x \| \mathsf{com}_M$, or if $\mathcal{A}$ ever makes two valid queries $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and $(\mathsf{com}_x, x', r'_x, \mathsf{com}_M, \tau_x)$ with $x \neq x'$, the simulator aborts.

Simulation of the protocol. The simulator sends $(\textsc{toss}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle)$ to functionality $\mathcal{F}_{\mathsf{coin}}$ and runs the protocol as an honest B through step 4. Upon receiving the message (t', v) (and assuming it does not abort the protocol), the simulator checks if $\mathcal{A}$ has made a valid query and aborts if not. Otherwise, it sends $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{A})$ to functionality $\mathcal{F}_{\mathsf{coin}}$ and obtains coins in return. It then uses the value x it has recorded, computes $y := \mathsf{coins} \oplus \mathsf{Ext}(x, v)$, and sends y as step 5 of the protocol. If $\mathcal{A}$ responds with $\tilde{x} = \mathsf{Ext}(x, v)$ and a valid proof of knowledge then Sim sends $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B} \rangle, \mathsf{B})$ to functionality $\mathcal{F}_{\mathsf{coin}}$; in any other case it simply aborts.

Let Real be the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of ψ between an honest B and $\mathcal{A}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the output of the honest B.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest B, functionality $\mathcal{F}_{\mathsf{coin}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. Here we imagine a real-world execution of the protocol between $\mathcal{A}$ (controlled by $\mathcal{Z}$) and an honest B, where at the end of the execution $\mathcal{Z}$ receives the output of B in addition to the final view of $\mathcal{A}$. However, the execution is aborted if $\mathcal{A}$ ever makes a valid query $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ to $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ but τ_x was not previously generated by B as a MAC tag on $\mathsf{com}_x \| \mathsf{com}_M$, or if $\mathcal{A}$ ever makes two valid queries $(\mathsf{com}_x, x, r_x, \mathsf{com}_M, \tau_x)$ and $(\mathsf{com}_x, x', r'_x, \mathsf{com}_M, \tau_x)$ with $x \neq x'$. Security of the MAC and binding of the commitment scheme imply that the view of $\mathcal{Z}$ here is indistinguishable from its view in Real.

Experiment $\mathcal{H}_2$. This experiment is identical to the previous one except that the experiment is also aborted if, in step 4, $\mathcal{A}$ sends (t', v) with $t' = t$, but $\mathcal{A}$ never made a valid query to $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$. Hiding of the commitment scheme, along with the fact that t is a uniform λ-bit string, implies that the view of $\mathcal{Z}$ here is indistinguishable from its view in experiment $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. Here, we modify the experiment to abort if the proof of knowledge succeeds in the final step of the protocol, yet $\tilde{x} \neq \mathsf{Ext}(x, v)$ (where x is the value used in the unique valid query made by $\mathcal{A}$). We prove that this has only a negligible effect on the view of $\mathcal{Z}$.

Let E be the above event, and let E' be the event that E occurs and in addition extraction of a witness from the argument of knowledge succeeds. The probabilities of E and E' are negligibly close. Let $\delta(\lambda)$ denote the probability of event E'. Imagine running the entire experiment twice using the same randomness except for choosing two independent values e, e'. The probability that E' occurs both times is at least $\delta(\lambda)^2$. There are now two possibilities:

1. In at least one of these executions, the witness extracted consists of values x', r'_x such that $\mathsf{Open}(\mathsf{com}_x, x', r'_x) = 1$ and $\tilde{x} = \mathsf{Ext}(x', v)$. In this case, $x' \neq x$ and so this violates binding of the commitment scheme. (Note that the valid input that $\mathcal{A}$ sent previously to $\mathcal{F}^{\mathsf{B}}_{\mathsf{wrap}}$ gives an opening to x.)

2. In the two executions, values M, r_M and M', r'_M are extracted with $\mathsf{Open}(\mathsf{com}_M, H(M), r_M) = 1$, $\mathsf{Open}(\mathsf{com}_M, H(M'), r'_M) = 1$, $M(\mathsf{com}_x, \mathsf{com}_M, \tau_x) = e$, and $M'(\mathsf{com}_x, \mathsf{com}_M, \tau_x) = e'$. So $M \neq M'$, but statistical binding of the commitment scheme implies that $H(M) = H(M')$, contradicting collision resistance.

We conclude that $\delta^2(\lambda)$, and hence $\delta(\lambda)$, must be negligible.

Experiment $\mathcal{H}_4$. This experiment is as before, except that instead of choosing uniform y the simulator chooses uniform coins and sets $y := \mathsf{coins} \oplus x$ (where x is the value used in the unique valid query made by $\mathcal{A}$). The view of $\mathcal{Z}$ is identical in experiments $\mathcal{H}_4$ and $\mathcal{H}_3$.

Since the view of $\mathcal{Z}$ in experiment $\mathcal{H}_4$ is the same as its view in Ideal, this completes the proof of Theorem 4 for a malicious A.

5.2.2 Corrupted B

We show a simulator Sim such that no non-uniform, PPT environment $\mathcal{Z}$ can distinguish (1) an execution between an honest A running ψ and the dummy adversary $\mathcal{B}$ (who simply forwards messages to/from $\mathcal{Z}$) in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world and (2) an execution involving an honest A, functionality $\mathcal{F}_{\mathsf{coin}}$, and Sim in the ideal world. The simulator Sim internally runs $\mathcal{B}$, forwarding messages to/from the environment $\mathcal{Z}$. In addition, Sim simulates the following interactions with $\mathcal{B}$.

Token. Sim waits to receive a message $(\textsc{create}, \langle \mathsf{sid}, \mathsf{B}, \mathsf{A}\rangle, M_{\mathcal{B}})$ from $\mathcal{B}$ to the $\mathcal{F}_{\mathsf{wrap}}$ functionality, and records $M_{\mathcal{B}}$.

Simulation of the protocol. Sim sends $(\textsc{toss}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle)$ to functionality $\mathcal{F}_{\mathsf{coin}}$. In step 2 it commits to $M_{\mathcal{B}}$, resulting in commitment com_M and decommitment r_M; it runs steps 3–5 honestly. Assuming it has not aborted the protocol, it sends $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, \mathsf{A})$ and $(\textsc{go}, \langle \mathsf{sid}, \mathsf{A}, \mathsf{B}\rangle, \mathsf{B})$ to functionality $\mathcal{F}_{\mathsf{coin}}$ and obtains coins in return. Then, in step 6, it sets $\tilde{x} := y \oplus \mathsf{coins}$ and gives a proof of knowledge using witness $M_{\mathcal{B}}, r_M$.

Let Real be the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution of ψ between an honest A and $\mathcal{B}$ in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid world. (We stress that the view of $\mathcal{Z}$ includes the view of $\mathcal{A}$ as well as the output of the honest A.) We show that this is computationally indistinguishable from Ideal, the distribution (ensemble) of the view of $\mathcal{Z}$ in an execution between an honest A, functionality $\mathcal{F}_{\mathsf{coin}}$, and Sim in the ideal world. To prove this, we consider a sequence of experiments as described next.

Experiment $\mathcal{H}_1$. This is similar to the real-world execution of the protocol, except that com_M is generated as a commitment to $M_{\mathcal{B}}$ rather than a commitment to 0^λ. Hiding of the commitment scheme immediately implies that the view of $\mathcal{Z}$ in experiment $\mathcal{H}_1$ is indistinguishable from its view in Real.

Experiment $\mathcal{H}_2$. This is identical to the previous experiment, except that now the proof of knowledge is done using witness $M_{\mathcal{B}}, r_M$ instead of witness x, r_x. Witness indistinguishability of the proof system implies that the view of $\mathcal{Z}$ here is indistinguishable from its view in experiment $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. Here, $\tilde{x}$ is chosen uniformly rather than being computed as $\tilde{x} := \mathsf{Ext}(x, v)$. We claim that the view of $\mathcal{Z}$ in experiment $\mathcal{H}_3$ is statistically indistinguishable from its view in $\mathcal{H}_2$. To see this, consider the information that $\mathcal{B}$ has about x. Since the commitment to x is statistically hiding, it reveals no information about x. The only information that $\mathcal{F}_{\mathsf{wrap}}$ can communicate to $\mathcal{B}$ is a single bit based on whether or not it aborts. Thus, with all but negligible probability, the min-entropy of x is at least 1.5λ, and so $\tilde{x}$ in $\mathcal{H}_2$ is statistically close to uniform.

Experiment $\mathcal{H}_4$. Now, instead of choosing $\tilde{x}$ uniformly, a uniform value $\mathsf{coins} \in \{0,1\}^\lambda$ is chosen and the simulator sets $\tilde{x} := y \oplus \mathsf{coins}$. It is clear that this does not change the view of $\mathcal{Z}$.

Since the view of $\mathcal{Z}$ in experiment $\mathcal{H}_4$ is identical to its view in Ideal, this completes the proof of Theorem 4.

Acknowledgments

We thank the anonymous reviewers for their careful and thorough reading of our paper, and for their helpful comments.

References

[1] M. Abdalla, D. Catalano, and D. Fiore. Verifiable random functions from identity-based key encapsulation. In *Advances in Cryptology—Eurocrypt 2009*, volume 5479 of *LNCS*, pages 554–571. Springer, 2009.

[2] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.

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

[4] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual Symposium on Foundations of Computer Science*, pages 186–195. IEEE, 2004.

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

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

[7] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001. Full version at `http://eprint.iacr.org/2000/067`.

[8] R. 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.

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

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

[11] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503. ACM Press, 2002.

[12] R. Canetti, R. Pass, and A. Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *48th Annual Symposium on Foundations of Computer Science*, pages 249–259. IEEE, 2007.

[13] N. Chandran, V. Goyal, and A. 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.

[14] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology—Crypto '92*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[15] R. Cramer and T. P. Pedersen. Improved privacy in wallets with observers. In *Advances in Cryptology—Eurocrypt '93*, volume 765 of *LNCS*, pages 329–343. Springer, 1993.

[16] I. Damgård, J. B. Nielsen, and D. 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.

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

[18] Y. Desmedt and J.-J. 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.

[19] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *8th Intl. Workshop on Theory and Practice in Public Key Cryptography (PKC)*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.

[20] N. Döttling, D. Kraschewski, and J. 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.

[21] N. Döttling, T. Mie, J. Müller-Quade, and T. Nilges. Implementing resettable UC-functionalities with untrusted tamper-proof hardware-tokens. In *10th Theory of Cryptography Conference—TCC 2013*, volume 7785 of *LNCS*, pages 642–661. Springer, 2013.

[22] M. Dubovitskaya, A. Scafuro, and I. Visconti. On efficient non-interactive oblivious transfer with tamper-proof hardware, 2010. Cryptology ePrint Archive, Report 2010/509.

[23] M. Fischlin, B. Pinkas, A.-R. Sadeghi, T. Schneider, and I. Visconti. Secure set intersection with untrusted hardware tokens. In *Cryptographers' Track—RSA 2011*, volume 6558 of *LNCS*, pages 1–16. Springer, 2011.

[24] O. Goldreich. *Foundations of Cryptography, vol. 2: Basic Applications*. Cambridge University Press, Cambridge, UK, 2004.

[25] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, 1989.

[26] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-time programs. In *Advances in Cryptology—Crypto 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, 2008.

[27] S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In *Advances in Cryptology—Crypto '92*, volume 740 of *LNCS*, pages 228–245. Springer, 1993.

[28] V. Goyal, Y. Ishai, M. Mahmoody, and A. Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Advances in Cryptology—Crypto 2010*, volume 6223 of *LNCS*, pages 173–190. Springer, 2010.

[29] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. 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.

[30] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology—Crypto '96*, volume 1109 of *LNCS*, pages 201–215. Springer, 1996.

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

[32] C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam. Composable security in the tamper-proof hardware model under minimal complexity. In *14th Theory of Cryptography Conference—TCC-B 2016*, volume 9985 of *LNCS*, pages 367–399. Springer, 2016. Prior versions available at `https://eprint.iacr.org/2015/887`.

[33] D. Hofheinz and T. Jager. Verifiable random functions from standard assumptions. In *13th Theory of Cryptography Conference—TCC-A 2016*, volume 9562 of *LNCS*, pages 336–362. Springer, 2016.

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

[35] S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. In *Advances in Cryptology—Eurocrypt 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, 2010.

[36] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology—Crypto 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, 2003.

[37] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer—efficiently. In *Advances in Cryptology—Crypto 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, 2008.

[38] K. Järvinen, V. Kolesnikov, A.-R. Sadeghi, and T. 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.

[39] J. 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.

[40] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, 2nd edition.* Chapman and Hall/CRC Press, 2014.

[41] J. Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31. ACM Press, 1988.

[42] V. 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.

[43] H. Lin, R. Pass, and M. Venkitasubramaniam. A unified framework for concurrent security: Universal composability from stand-alone non-malleability. In *41st Annual ACM Symposium on Theory of Computing*, pages 179–188. ACM Press, 2009.

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

[45] H. K. Maji, M. Prabhakaran, and M. Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In *6th Theory of Cryptography Conference—TCC 2009*, volume 5444 of *LNCS*, pages 256–273. Springer, 2009.

[46] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130. IEEE, 1999.

[47] T. Moran and G. 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.

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

A A Technical Result

We consider the following *dot-product-with-equality-oracle game* between two parties A and T. First, a uniform bit b is chosen. Values $z, h \in \mathbb{F}_2^{4\lambda}$ are then chosen uniformly subject to $z^T h = b$ and h nonzero; A is given h, and T is given z. Parties A and T then interact with a third party for at most $p(\lambda)$ rounds; in round j, they may send arbitrary inputs a_j and t_j to this third party, who responds to both parties with a bit indicating whether a_j and t_j are equal. If so, the parties continue to the next round, but once $a_j \neq t_j$ the protocol ends. (Otherwise, it ends in round $p(\lambda)$.) When the protocol ends, A outputs a guess for b and succeeds if its guess is correct. The advantage of A is the absolute value of its probability of success minus 1/2. We claim:

Theorem 5 *For any (even all-powerful) A and T and any polynomial p, the advantage of A in the dot-product-with-equality-oracle game is negligible.*

Proof Fix A, T, and p, and assume that for infinitely many values of λ, the probability that A succeeds is $1/2 + 1/q(\lambda)$. In what follows we restrict our attention to such λ only. We also assume A and T are deterministic without loss of generality.

We introduce a second, one-shot game that proceeds as follows. Parties A' and T' are given h and z, respectively, where h, z are generated as before, and then each party sends a single input $(a_1, \ldots, a_p)$ and $(t_1, \ldots, t_p)$ to a third party. That third party computes $f((a_1, \ldots, a_p), (t_1, \ldots, t_p))$, where f returns the first index i where $a_i \neq t_i$ (which can be encoded as a string of length $\log p$.) Then A' outputs a guess for b and succeeds if its guess is correct.

We claim that there exist A' and T' such that A' succeeds with probability $1/2 + 1/q(\lambda)$ in this second game. T' is constructed by internally running T, simulating the behavior of the equality check by simply returning 1 for all p rounds. Let $\mathbf{t} = (t_1, \ldots, t_p)$, where t_i is the input that T sends to its (simulated) third party in round i. Then T' sends $\mathbf{t}$ to its third party. A' is constructed similarly, resulting in an input $\mathbf{a} = (a_1, \ldots, a_p)$ that is sent to the third party. After learning the index i from the third party, A' again runs A, but this time simulating the third party by returning 1 for the first $i - 1$ rounds and then returning 0 in round i and ending the protocol. Finally, A' outputs whatever A does. It is immediate that A' succeeds exactly when A does.

We now consider a third game in which A'' gets no input and T'' is given uniform input z. Party A'' may send a list of inputs $(\mathbf{a}^1, \ldots, \mathbf{a}^\ell)$ to a third party, where $\mathbf{a}^j = (a_1^j, \ldots, a_p^j)$, and T'' sends a single input $\mathbf{t} = (t_1, \ldots, t_p)$. The third party returns $f(\mathbf{a}^1, \mathbf{t}), \ldots, f(\mathbf{a}^\ell, \mathbf{t})$ to A'', who then outputs a guess for z and succeeds if this guess is correct.

By adapting the proof of the Goldreich-Levin theorem [25] in a straightforward manner (viewing $z^T h$ as a "hard-core bit" guessed with probability noticeably better than $1/2$ by A'), we can conclude that there exist A'', T'', and polynomials ℓ, q'' such that A'' succeeds with probability at least $1/q''(\lambda)$ in this third game.

We next observe that the answer of the third party in the third game can be *compressed.* Let $\mathbf{a}^j = (a^j_1, \ldots, a^j_p)$ and $\mathbf{t} = (t_1, \ldots, t_p)$. Then for each j, the third party can compute i_j, the smallest index for which $a^j_{i_j} \neq t_{i_j}$; it then returns (j^*, i_{j^*}), where j^* maximizes i_{j^*} (with ties broken arbitrarily). Given (j^*, i_{j^*}) and all the $\mathbf{a}^j$, it is possible to reconstruct $f(\mathbf{a}^1, \mathbf{t}), \ldots, f(\mathbf{a}^\ell, \mathbf{t})$.

The answer in this modified game has length $\log \ell + \log p$, where ℓ and p are polynomial, and thus can be guessed with non-negligible probability. But this implies guessing a uniform input $z \in \mathbb{F}_2^{4\lambda}$ with non-negligible probability, something that is clearly impossible. ■