Equational Security Proofs of Oblivious Transfer Protocols[*]

Baiyu Li[†] Daniele Micciancio[‡]

June 15, 2016

Abstract

We exemplify and evaluate the use of the equational framework of Micciancio and Tessaro (ITCS 2013) by analyzeing a number of concrete Oblivious Transfer protocols: a classic OT transformation to increase the message size, and the recent (so called "simplest") OT protocol in the random oracle model of Chou and Orlandi (Latincrypt 2015), together with some simple variants. Our analysis uncovers subtle timing bugs or shortcomings in both protocols, or the OT definition typically employed when using them. In the case of the OT length extension transformation, we show that the protocol can be formally proved secure using a revised OT definition and a simple protocol modification. In the case of the "simplest" OT protocol, we show that it cannot be proved secure according to either the original or revised OT definition, in the sense that for any candidate simulator (expressible in the equational framework) there is an environment that distinguishes the real from the ideal system.

Keywords Oblivious transfer, secure multiparty computation, universal composability, equational security, asynchronous, simulation-based

1 Introduction

Cryptographic design and analysis is a notoriously hard problem, arguably even harder than standard software design because it requires to build systems that behave robustly in the presence of a malicious adversary that actively tries to subvert their execution. The desirability of precise formalisms to describe and analyze cryptographic constructions is well exemplified by the code-based game-playing framework of [BR06] to present security definitions and proofs of standard cryptographic functions. But even the detailed framework of [BR06] offers little help when formalizing more complex cryptographic protocols, due to their interactive nature and underlying distributed execution model. At the semantic level, the gold standard in secure computation protocol design and analysis is the *universally composable (UC)* security model of [Can01] (or one of its many technical variants [CLOS02, BCNP04, Küs06, BPW07, CDPW07, HUM13, HS15],) which offers strong compositionality guarantees in fully asynchronous execution environments like the Internet. Unfortunately, the relative lack of structure/abstraction in the traditional formulation of this model[1] makes it rather hard to use in practice, when specifying and analyzing concrete protocols.[2] These limitations are widely recognized, and have prompted researchers to explore several variants, simplifications and specialization of the general UC security model [KT13, KMTZ13, CCL15, Wik16]. In this perspective, a very interesting line of work is represented by the "abstract cryptography" framework of [MR11], which calls for an axiomatic

[*]This work was supported in part by NSF grant CNS-1528068. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

[†]University of California, San Diego, USA. E-mail: `baiyu@cs.ucsd.edu`

[‡]University of California, San Diego, USA. E-mail: `daniele@cs.ucsd.edu`.

[1]Rooted in computational complexity, the model is usually described as an arbitrary network of (dynamically generated) Turing machines that communicate by means of shared tapes, possibly under the direction of some scheduling process, also modeled as an interactive Turing machine.

[2]This is analogous to the Turing machine, an excellent model to study computation in general but a rather inconvenient one when it comes to specifying actual algorithms.

approach to the description and analysis of cryptographic primitives/protocols, and the "constructive cryptography" [Mau12] and "equational security" [MT13] frameworks, which can be thought of as logical models of the axioms put forward in [MR11].

In this work we examine the equational security framework of [MT13], which provides both a concrete mathematical model of computation/communication, and a concise syntax to formally describe distributed systems by means of a set of mathematical equations. We believe that progress in our ability to describe and analyze cryptographic protocols cannot be achieved simply by formulating frameworks and proving theorems in definitional papers, but it requires putting the frameworks to work on actual example protocols. To this end, we present a detailed case-study where we evaluate the expressiveness and usability of this framework by analyzing a number of concrete *oblivious transfer* protocols, a simple but representative type of security protocols of interest to cryptographers.

Oblivious transfer (OT), in its most commonly used 1-out-of-2 formulation [EGL85], is a two party protocol involving a sender transmitting two messages m_0, m_1 and a receiver obtaining only one of them m_b, in such a way that the sender does not learn which message $b \in \{0, 1\}$ was delivered and the receiver does not learn anything about the other message m_{1-b}. OT is a classic example of secure computation [Rab81, EGL85], and an important (in fact, complete) building block for the construction of arbitrary security protocols [Yao86, GMW87, Kil88, CGT95, IPS08, LP11]. In Sections 3 and 4 we investigate a well known transformation often used to increase the message length of OT protocols with the help of a pseudorandom generator. In Section 5, we investigate a very efficient OT protocol in the random oracle model recently proposed in [CO15].

We remark that the primary goal of our work is to exemplify and evaluate the usability of the equational security framework of [MT13], rather than finding and fixing bugs in specific protocol instances. Still, our findings about the OT protocols under study may be of independent interest, and well illustrate how equational security modeling can offer a convenient and valuable tool for cryptographic protocol specification and analysis. The main findings about the OT protocols are the following:

- The security of the OT protocol transformation, often considered a folklore result in cryptography, does not hold with respect to the naive OT definition typically used (often implicitly) in the cryptographic literature. However, if the OT ideal functionality definition is suitably modified, then the transformation becomes provably secure, and can be readily analyzed using simple equational reasoning.
- The protocol of [CO15] can be proved secure according to *neither* the classic *nor* the revised OT definitions considered above. We also consider a weaker OT definition with the hope to prove security, but as we will show the protocol is still not secure in that model.

Technical details about our findings, and general comments/conclusions are provided in the next paragraphs.

1.1 Oblivious Transfer Extension.

The standard definition of OT is given by a functionality $\mathsf{OT}((m_0, m_1), b) = m_b$ that takes a pair of messages (m_0, m_1) from the sender, a selection bit b from the receiver, gives m_b to the receiver, and gives nothing to the sender. The two messages are assumed to have the same length $|m_0| = |m_1| = \kappa$, which is usually tied to the security parameter of the scheme and the mathematical structures used to implement it. (E.g., $\kappa = \log |H|$ where H is the domain/range of some group-theoretic cryptographic function.) A natural and well known method to adapt such OT protocol to one allowing the transmission of longer messages is the following:

1. Use an underlying OT protocol to send two random seeds (s_0, s_1) of length κ,
2. Use these seeds as keys to encrypt the two messages using a private-key encryption scheme,[3] and send both ciphertexts to the receiver over a standard (authenticated, but insecure to eavesdropping) communication channel.

[3]Since each seed s_i is used only once, the secret key encryption scheme can be as simple as stretching s_i using a pseudorandom generator $\mathcal{G}$, and use the resulting string $\mathcal{G}(s_i)$ as a one-time pad to mask the message m_i.

The intuition is that since the receiver gets only one of the two keys, the other message is protected by the encryption scheme. Indeed, the intuition is correct, in the sense that encryption does its job and protects the other message, but the protocol is nevertheless not secure (at least, according to the simulation-based fully asynchronous security definition implied by the OT functionality described above.) Our formal analysis shows that, while the protocol is correct, and secure against corrupted senders, it is not secure against corrupted receivers, and for a very simple reason: it contains a subtle timing bug! In a real execution, the sender transmits the encryption of its two messages as soon as the two messages are made available by the environment. However, the simulator can produce the corresponding simulated ciphertexts only after the receiver has chosen its selection bit b. In order to prove security, the sender should delay the transmission of the ciphertexts until after the receiver has provided b to the underlying OT protocol. The problem is that the above OT ideal functionality does not disclose any information to the sender, not even if and when the receiver has selected the bit b.

We also consider a revised OT definition $\mathsf{OT}((m_0, m_1), b) = (f(b), m_b)$, that includes an additional output $f(b) \in \{\perp, \top\}$ disclosing to the sender if b has been chosen yet, without providing the actual value of $b \in \{0, 1\}$. We modify the protocol accordingly (by letting the sender delay the transmission of the ciphertexts until $b > \perp$), and show that the modified protocol can be formally proved secure according to the revised OT definition.

1.2 OT in the Random Oracle Model.

In [CO15], Chou and Orlandi propose a new OT protocol achieving UC security in the random oracle model [BR93]. The protocol is very elegant and can be efficiently implemented based on elliptic curve groups. We provide a formal analysis of the protocol using the equational framework. We show that if the naive OT definition is used, then the protocol is secure against corrupted receivers, but it is not secure against corrupted senders. The failure of the simulator is due to the fact that in a real protocol execution the sender learns if and when the receiver provides its selection bit b. As in the analysis of the OT length extension transformation, this problem can be addressed by switching to the revised OT definition given above, and modifying the sender program in the obvious way. However, this time, changing the definition breaks the proof of security against corrupted receivers. In fact, we formally show that (with respect to the revised OT definition) when the receiver is corrupted, for every candidate simulator there exists an (efficient) environment that distinguishes the real system from the ideal system with non-negligible advantage.

1.3 Discussion/Conclusions

Before jumping to conclusions, some remarks about the significance of our results are in order. As already noted, it should be understood that the aim of our work was to illustrate the use of the equational framework, rather than criticizing any specific protocol or definition. In particular, we are not arguing that the revised OT definition given in Section 4 is the "correct" one, and everybody should use it. In fact, other alternative definitions are possible. Our main point is that the equational model is a convenient framework to precisely formulate and investigate alternative definitions.

The OT message length transformation studied in Section 3 is folklore. We are not aware of any work analyzing its security, and our study is, to the best of our knowledge, the first work even making a formal security claim about it. This is perhaps because doing this using the traditional framework based on the informal use of interactive Turing machines already seemed cumbersome and error prone enough not be worth the effort. In fact, the transformation is simple enough that at first it is natural to wondered if a formal proof of security is required at all. Our analysis shows that a formal security proof is indeed useful, at very least to unambiguously identify the security property (ideal functionality) for which the transformation is (proved or claimed to be) correct. We remark that when we set to analyze the OT protocol transformation, we were giving for granted that the transformation was secure, and the analysis was meant primarily as a simple example to illustrate the use of the equational framework. Finding that the protocol does not emulate the traditional OT definition came to us as a surprise, even if in hindsight the timing bug is rather obvious.

In this respect, the equational framework proved to be a very convenient tool to carry out a precise formal analysis with relatively modest effort.

As for the protocol of [CO15], we are not claiming it should or should not be used as it is. We are certainly not concerned about whether the protocol is making a "morally correct" use of the random oracle, or if "global" random oracle definition [CJS14] should be used instead. We simply use the equational framework to model and analyze the protocol as described in the original paper [CO15]. We had noticed that this work appeared to use the traditional OT definition, and our original goal was to show that the protocol does not satisfy it, but it can be proved secure if the OT ideal functionality is revised as in the OT transformation case. We still find it rather surprising that the protocol cannot be proved secure according to either definition. We remark that what we mean by "cannot be proved secure" is that for any candidate simulator (within the model) there exists a distinguishing environment that can tell the real system and ideal system apart. We do not have a direct attack to the protocol. So, by no means our results should be interpreted as a cryptanalysis of [CO15], and, in fact, we believe that the protocol provides some meaningful form of security. Also, our results do not point to any specific bug in the (informal) proof in the original paper [CO15], which, most likely, can be properly interpreted as a proof of security in some kind of semi-synchronous model where the environment is required to provide inputs to all parties at the same time. However, we do not consider this a satisfactory answer because the semi-synchronous interpretation does not provide a truly composable notion of security,[4] as one would normally expect in the UC model.

We believe our analysis highlights the importance of a more rigorous proof style when analyzing secure computation protocols than currently feasible using traditional formulations of the UC framework and its variants. This is especially important when it comes to formally specifying the security properties satisfied (or claimed) by a protocol. Without an unambiguous formal security specification/claim, even the most detailed proof is of little value, as it is not clear what is being proved or claimed. Withing the context of our work, the equational framework of [MT13] proved to be a very convenient and useful formalism to express security definitions (in the form of ideal functionalities) and cryptographic protocols in a concise, yet mathematically precise way. It allowed to easily explore different definitional variants and put them to good use to spot potential bugs in cryptographic protocols. While the equational framework proved to be more than adequate to describe concrete cryptographic protocols and the security definitions they achieve, it is less clear if it powerful enough to express any conceivable simulation strategy. In particular, an intriguing interpretation of our negative security results about the protocol of [CO15] is that perhaps the protocol is secure, but the simulator (demonstrating its security) cannot be expressed within the framework of [MT13]. If this is the case, it would be very interesting to identify a suitable extension of the model of [MT13] that allows to formally analyze the protocol of [CO15] with respect to an appropriate security definition. Exploring the applicability of abstract frameworks along the lines of [MR11, Mau12, MT13] to the specification and analysis of a wider range of cryptographic protocols is likely to be mutually beneficial, both to further develop and refine the models, and to gain useful insight on the security of concrete cryptographic protocols.

2 Background and Notations

In this section we review the equational framework of [MT13], and define the notation used in this paper. We recall that the execution model of [MT13] consists of a network, with nodes representing computational units, and (directed) edges modeling communication channels. (See below for details.) Each channel is associated with a partially ordered set of channel "histories" or "behaviors", representing all possible messages or sequences of messages that may be transmitted on the channel over time. The partial order relation represents temporal evolution, so for any two histories $h_1 \leq h_2$ means that h_2 is a possible extension (or future) of h_1. The standard example is that of finite sequences $M^* = \{(m_1, \ldots, m_k) : k \geq 0, \forall i. m_i \in M\}$ of messages from a ground set M, ordered according to the prefix partial ordering relation. Another common example,

[4]The problem is that, when the OT protocol is used within part of a larger protocol, even if the users provide all their inputs at the same time, different parties may still engage in the execution of sub-protocols at different times, depending on message scheduling and interleaving with other protocols, possibly under the influence of an adversary.

modeling a channel capable of delivering only a single message, is the flat partial order $M_\perp$, consisting of all messages in M and a special element $\perp$ denoting the fact that no message has been transmitted yet. Different incoming and outgoing channels (incident to a single node) are combined taking Cartesian products, so that each node can be thought as having just one input and one output. The computational units at the nodes are modeled as functions $f\colon C_1 \to C_2$ from the incoming channels to the outgoing channels, satisfying the natural monotonicity requirement that for any $h_1 \leq h_2$ in C_1, we have $f(h_1) \leq f(h_2)$ in C_2. Informally, monotonicity captures the intuition that once a party transmits a message, it cannot go back in time and take it back. The main advantage of the equational framework is that it has a mathematically clean and well defined semantics, where processes (nodes) can be described by simple mathematical equations (specifying the relation between the input and the output of the node), and composition simply combines equations together. In particular, composition is independent of the composition order, and the behavior of a network is fully specified by the equations of each node, and the edges connecting the nodes. Using equations to describe systems also provides a simple and precise way to reason about transformations as well. For example, equivalent components (in the sense of having equivalent equations) can be replaced by each other, and when considering probabilistic behaviors, if a component is indistinguishable from another component, then they can be used interchangeably with negligible impact on the behavior of the entire system.

For completeness, in the next paragraph we will give some background on the (standard) theory that gives a precise meaning to systems of equations as used in [MT13] and in this paper. This material is important to give a solid mathematical foundation to the equational framework, but is not essential to follow the rest of the paper, and the reader may want to skip directly to the following paragraph describing our notational conventions.

Complete Partial Orders (CPOs) The mathematical foundation of the equational framework is provided by domain theory, which is the standard tool used in functional programming language design for giving precise mathematical meanings to expressions. Here we give just enough background to describe the systems studied in this paper, and refer the reader to [Sco82, GS90, SHLG94] for a detailed treatment. Recall that a partially ordered set (or poset) is a set X equipped with a reflexive, transitive and antisymmetric relation $\leq$. All posets considered in this paper are complete partial orders (CPOs), i.e., any (possibly empty) chain $x_1 < x_2 < \ldots$ has a least upper bound $\sup_i x_i$. These posets are endowed with the *Scott topology*, where a subset $C \subseteq X$ is *closed* if for all $x \in C$, $y \leq x$ implies $y \in C$, and any chain in C has a least upper bound in C. *Open* sets are defined in the usual way as the complement of closed sets. The standard topological definition of *continuous function* still applies here, and continuous functions (with respect to the Scott topology) are exactly the functions that preserve limits $f(\sup_i x_i) = \sup_i f(x_i)$. The set of all continuous functions from CPOs X to Y is denoted by $[X \to Y]$. Notice that any (Scott) continuous function is necessarily *monotone*, i.e., for all $x, y \in X$, if $x \leq y$ then $f(x) \leq f(y)$. All CPOs X have a minimal element $\perp = \sup \emptyset$, which satisfies $\perp \leq x$ for all $x \in X$. For any set A, we can always construct a *flat* CPO by extending A with a unique *bottom* element $\perp$, and we write such CPO as $A_\perp = A \cup \{\perp\}$. The partial ordering in this flat CPO consists of $\perp \leq x$ for all $x \in A$. It should be easy to see that all nonempty closed sets in $A_\perp$ contain $\perp$, and open sets in $A_\perp$ are exactly the subsets of A and the whole $A_\perp$.

Unless otherwise noted, all CPOs we consider are flat partial orderings $X_\perp$ for some finite set X, and functions $f\colon X \to Y$ between sets are lifted to strict functions $f\colon X_\perp \to Y_\perp$ between the corresponding flat CPOs by setting $f(\perp) = \perp$. The bottom element usually designates the situation where no (real) input or output is given yet. The Cartesian product $X \times Y$ of two CPOs is a CPO with the component-wise partial order $(x_1, y_1) \leq (x_2, y_2) \iff x_1 \leq x_2 \wedge y_1 \leq y_2$.

For any CPO X, every continuous functions $f\colon X \to X$ admits a *least fixed point*, which is the minimal $x \in X$ such that $f(x) = x$. The least fixed point can be obtained by taking the limit of the sequence $\perp, f(\perp), f^2(\perp), \ldots$. We can use least fixed points to find the solution to a system of mutually recursive equations. Such solution describes the final outputs of interactive computations between nodes in a network. The least fixed points can also be used to simplify combined systems. For example, if in a system we have an equation $f(z) = g(z, x)$ for continuous f and g, and in another system we have $x = h(z, x)$ for continuous h, then to combine these two systems we can find the least fixed point x_z to the function $h_z(x) = h(z, x)$

and then simplify the combined system as $f(z) = g(z, x_z)$.

To study cryptographic protocols we must be able to specify and analyze their probabilistic behaviors. In the equational framework, probabilistic functions are simply (continuous) functions between sets of distributions with the appropriate ordering relation. A *probability distribution* on a CPO X is a function $p\colon X \to [0,1]$ such that[5] $p(A) + p(B) = p(A \cup B)$ for all disjoint $A, B \subseteq X$ and $p(X) = 1$. As usual, we say that a probability p is *negligible* if for all $x \in X$, $p(x) < n^{-c}$ for any constant $c > 1$, where n is a *security parameter*.[6] Similarly, p is *overwhelming* if $1 - p$ is negligible. If X is a CPO, then the *set of probability distributions over* X, denoted by $D(X)$, is also a CPO, where for any two distributions $p \le q$ (in $D(X)$) if and only if $p(A) \le q(A)$ for any open subset $A \subseteq X$.

Notation In this paper we mostly use flat CPOs, i.e., partially ordered sets $\mathbb{X}$ with a bottom element $\perp \in \mathbb{X}$ such that $x_1 \le x_2$ iff $x_1 = \perp$ or $x_1 = x_2$. These are used to model simple communication channels that can transmit a single message from $\mathbb{X} \setminus \{\perp\}$, with $\perp$ representing the state of the channel before the transmission of the message. For any CPO $\mathbb{X}$, we write $\mathbb{X}^{\times 2} = \{(x,y)\colon x, y \in \mathbb{X}, x \ne \perp, y \ne \perp\}_\perp$ for the set of *strict* pairs over $\mathbb{X}$. The elements of a pair $z \in \mathbb{X}^{\times 2}$ are denoted $z[0]$ and $z[1]$, with $z[i] = \perp$ when $z = \perp$ or $i = \perp$. The operation of combining two elements into a strict pair is written $\langle x, y\rangle$. Notice that $\langle x, \perp\rangle = \langle \perp, y\rangle = \perp$, and therefore $\langle x, \perp\rangle[0] = \langle \perp, y\rangle[1] = \perp$ even when $x, y \ne \perp$. For any set A, we write $x \leftarrow A_\perp$ for the operation of selecting an element $x \ne \perp$ uniformly at random from A.

It is easily verified that for any pairs $z, \langle x_0, x_1\rangle, \langle y_0, y_1\rangle$, strict function f and strict binary operation $\odot$,

$$\begin{aligned} z &= \langle z[0], z[1]\rangle && (1)\\ f(\langle x_0, x_1\rangle[i]) &= \langle f(x_0), f(x_1)\rangle[i] && (2)\\ \langle x_0, x_1\rangle[i] \odot \langle y_0, y_1\rangle[i] &= \langle x_0 \odot y_0, x_1 \odot y_1\rangle[i] && (3)\end{aligned}$$

We use the following abbreviations for common CPOs and operations:

- The CPO $\mathbb{T} = \{\top\}_\perp$, representing *signals*, i.e., messages with no information content.
- The CPO $\mathbb{B} = \{0,1\}_\perp$ of single bit messages, often used to select an element from a pair.
- The CPO $\mathbb{M}_n = \{0,1\}^n_\perp$ of bit-strings of length n.
- $x!y = \langle x, y\rangle[1]$, the operation of *guarding* an expression y by some other expression x. Notice that $x!y = y$, except when $x = \perp$, and can be used to "delay" the transmission of y until after x is received.
- $x! = x!\top$, testing that $x > \perp$.

As an example, using the notations introduced so far, we can describe the ideal (1-out-of-2 Oblivious Transfer) OT functionality by the equations in Figure 1. (Notice that this functionality is parameterized by a message space $\mathbb{M}$.) The first line specifies the names of the functionality (OT), input channels (m_2, b) and output channel(s) m. This is followed by a specification of the type of each channel: the input interface includes a message pair $m_2 = \langle m_0, m_1\rangle \in \mathbb{M}^{\times 2}$ from a sender and a selection bit $b \in \mathbb{B}$ from a receiver. The output interface is a single message $m \in \mathbb{M}$ sent to the receiver while the sender does not get any information from the functionality. The last line $m = m_2[b]$ is an equation specifying the value of the output channel(s) as a function of the input channels. The functionality is illustrated by a diagram showing the names of the function and the input/output channels.

In the rest of this paper, equational variables usually belong to unique domains (e.g., $m_2 : \mathbb{M}_n^{\times 2}$.) So from now on, we will omit such type specifications when defining functions using equations, and we will follow the convention listed in Table 1 for naming variables.

[5]In general we should consider the Borel algebra on X when defining probability distributions on X. Here we simply use X instead since we work on finite sets and discrete probabilities.

[6]In the asymptotic setting, cryptographic protocols are parameterized by a security parameter n. For notational simplicity, we consider this security parameter n as fixed throughout the paper.

$$\begin{array}{rcl} \mathsf{OT}_{\mathbb{M}}(m_2, b) & = & m \\ m_2 & : & \mathbb{M}^{\times 2} \\ b & : & \mathbb{B} \\ m & : & \mathbb{M} \\ m & = & m_2[b] \end{array}$$

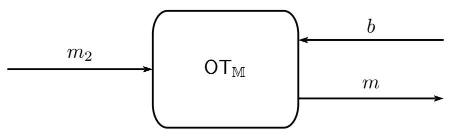


Figure 1: A naive Oblivious Transfer functionality: the receiver gets the selected message $m = m_2[b]$, and the sender does not get anything at all.

The definition of security in the equational framework follows the well-accepted simulation-based security paradigm. In this paper we consider only oblivious transfer protocols, which are two-party protocols between a sender program and a receiver program. So, the ideal functionality F is usually a function from $X_0 \times X_1$ to $Y_0 \times Y_1$, where X_i, Y_i are external input and output channels of party i in the protocol. The protocol consists of a pair of programs (functions) P_0, P_1, which, when combined with a communication network N, result in a system $(P_0|N|P_1)\colon X_0 \times X_1 \to Y_0 \times Y_1$ equivalent to the ideal functionality F. When a party P_i is *corrupted*, the adversary plays the role of P_i, and the resulting real system consists of just the remaining honest party and the network. A protocol is secure against the corruption of P_0 if there is an efficient simulator S_0 such that the systems $(N|P_1)$ and $(S_0|F)$ are computationally indistinguishable. Security against corruption of P_1 is defined similarly.

A distinctive feature of the equational framework is the ability to specify fully asynchronous systems. An adversary/environment might not provide all the input to a real system at the same time. So when analyzing the security of a protocol we must consider such asynchronous environments.

3 Oblivious Transfer Length Extension: a first attempt

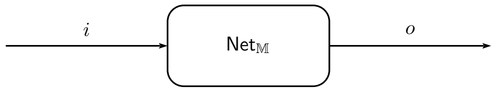
As an abbreviation, when the message space $\mathbb{M} = \{0,1\}^n_\perp$ is the set of all bitstrings of length n, we write OT_n instead of $\mathsf{OT}_{\mathbb{M}}$. Consider the following Oblivious Transfer *length extension* problem: given an OT_n channel for messages of some (sufficiently large) length n, build an oblivious transfer functionality OT_ℓ for messages of length $\ell > n$. The goal is to implement OT_ℓ making a single use of the basic OT_n functionality, possibly with the help of an auxiliary (unidirectional, one-time) communication channel for the transmission of messages from the sender to the receiver. For simplicity,[7] we model the communication channel as a functionality $\mathsf{Net}_{\mathbb{M}}$ that copies its input to the output

[7]This corresponds to a perfectly secure communication channel. More complex/realistic communication channels are discussed at the end of this section.

Variable name	Domain	Variable name	Domain
m	$\mathbb{M}_n$	m'	$\mathbb{M}_\ell$
m_2	$\mathbb{M}_n^{\times 2}$	m'_2	$\mathbb{M}_\ell^{\times 2}$
c_0,c_1	$\mathbb{M}_\ell$	c_2	$\mathbb{M}_n^{\times 2}$
a,a'	$\mathbb{T}$	b,b'	$\mathbb{B}$
i,o	$\mathbb{M}_n$	i_2,o_2	$\mathbb{M}_\ell^{\times 2}$
k	$\mathbb{K}_n$	k_2	$\mathbb{K}_n^{\times 2}$
q	$(G^2 \times G)_\perp$	q_2	$(G^2 \times G)_\perp^{\times 2}$
X,Y	$G_\perp$		

Table 1: Frequently used variables and their domains.

$$\begin{array}{rcl} \mathsf{Net}_{\mathbb{M}}(i) & = & o \\ i & : & \mathbb{M} \\ o & : & \mathbb{M} \\ o & = & i \end{array}$$



where the set $\mathbb{M}$ is an arbitrary message space. As before, when $\mathbb{M} = \{0,1\}^n_\perp$, we simply write Net_n instead of $\mathsf{Net}_{\mathbb{M}}$. The OT length extension protocol is specified by a pair of Sender and Receiver programs, which are interconnected (using the OT_n and $\mathsf{Net}_{2\ell}$ functionalities) as shown in Figure 2. Notice how the external input/output interface of the system corresponding to a real execution of the protocol in Figure 2 is the same as that of the ideal functionality $\mathsf{OT}_\ell(m'_2, b') = m'$ the protocol is trying to implement.

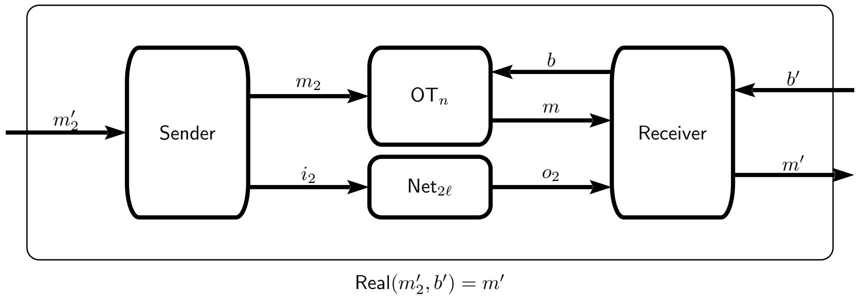
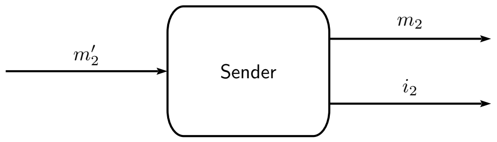


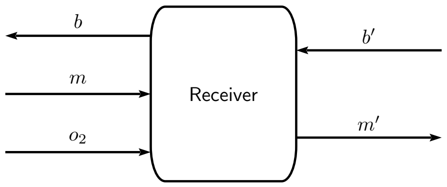
Figure 2: A real execution of a candidate OT length extension protocol. The protocol consists of a Sender and a Receiver programs that communicate using OT_n and $\mathsf{Net}_{2\ell}$ functionalities.

A natural approach to design an OT length extension protocol is to make use of a pseudorandom generator $\mathcal{G} : \mathbb{M}_n \to \mathbb{M}_\ell$ that stretches a short random seed of length n into a long pseudorandom string of length ℓ. Using such pseudorandom generator, one may define candidate Sender and Receiver programs as follows:

$$\begin{array}{rcl} \mathsf{Sender}(m'_2) & = & (m_2, i_2) \\ m_0 & \leftarrow & \mathbb{M}_n \\ m_1 & \leftarrow & \mathbb{M}_n \\ m_2 & = & \langle m_0, m_1 \rangle \\ c_0 & = & m'_2[0] \oplus \mathcal{G}(m_0) \\ c_1 & = & m'_2[1] \oplus \mathcal{G}(m_1) \\ i_2 & = & \langle c_0, c_1 \rangle \end{array}$$



$$\begin{array}{rcl} \mathsf{Receiver}(m, o_2, b') & = & (b, m') \\ b & = & b' \\ m' & = & o_2[b'] \oplus \mathcal{G}(m) \end{array}$$



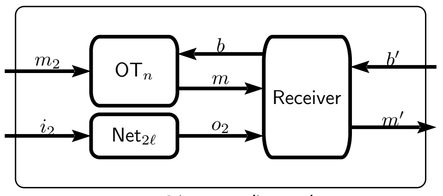
In words, these programs work as follows:

- The sender picks two random seeds m_0, m_1, and passes (one of) them to the receiver using the OT_n functionality. It then stretches the two seeds using the pseudorandom generator $\mathcal{G}$, and uses the generator's output as a one-time pad to "mask" the actual messages before they are transmitted to the receiver over the communication channel $\mathsf{Net}_{2\ell}$.

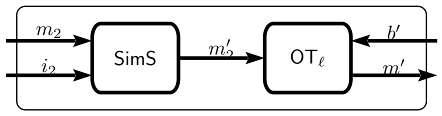
- The receiver selects one of the two seeds from the OT_n functionality, expands it using the pseudorandom generator, and uses the result to "unmask" the corresponding message from $\mathsf{Net}_{2\ell}$.

It is easy to show that the protocol is correct, in the sense that combining the equations of OT_n, $\mathsf{Net}_{2\ell}$, Sender and $\mathsf{Receiver}$ as shown in Figure 2 results in a system $\mathsf{Real}(m_2', b') = m'$ that is perfectly equivalent to the defining equation $m' = m_2'[b']$ of the ideal functionality OT_ℓ. Intuitively, the protocol also seems secure because only one of the two seeds can be recovered by the receiver, and the unselected message is protected by an unpredictable pseudorandom pad. But security of cryptographic protocols is a notoriously tricky business, and deserves a closer look.

We first consider the security of the protocol when the sender is corrupted. The attack scenario corresponds to the real system obtained by removing the Sender program from the protocol execution in Figure 2. Following the simulation paradigm, security requires exhibiting an efficient simulator program SimS (interacting, as a sender, with the ideal functionality OT_ℓ) such that the following real and ideal systems are computationally indistinguishable:



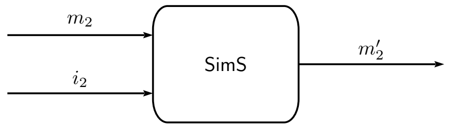
$\mathsf{RealS}(m_2, i_2, b') = m'$



$\mathsf{IdealS}(m_2, i_2, b') = m'$

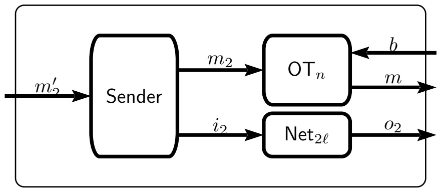
Security is easily proved by defining the following simulator:

$$\begin{aligned} \mathsf{SimS}(m_2, i_2) &= m_2' \\ m_0 &= i_2[0] \oplus \mathcal{G}(m_2[0]) \\ m_1 &= i_2[1] \oplus \mathcal{G}(m_2[1]) \\ m_2' &= \langle m_0, m_1 \rangle \end{aligned}$$

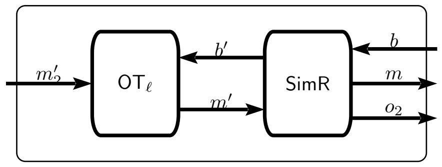


We observe that RealS and IdealS are perfectly equivalent because they both simplify to $m' = i_2[b'] \oplus \mathcal{G}(m_2[b'])$. So, the protocol is perfectly secure against corrupted senders.

We now turn to analyzing security against a corrupted receiver. This time we need to come up with a simulator SimR such that the following real and ideal executions are equivalent:



$\mathsf{RealR}(m_2', b) = (m, o_2)$



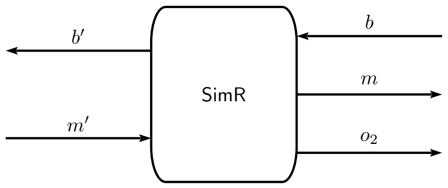
$\mathsf{IdealR}(m_2', b) = (m, o_2)$

Of course, this time we can only aim at proving computational security, i.e., coming up with a simulator such that RealR and IdealR are computationally indistinguishable. We begin by writing down explicitly the equations that define the real system execution. Combining the equations for Sender, OT_n and $\mathsf{Net}_{2\ell}$, we obtain the following system:

$$\begin{array}{rcl}
\mathsf{RealR}(m_2', b) & = & (m, o_2) \\
m_0 & \leftarrow & \mathbb{M}_n \\
m_1 & \leftarrow & \mathbb{M}_n \\
c_0 & = & m_2'[0] \oplus \mathcal{G}(m_0) \\
c_1 & = & m_2'[1] \oplus \mathcal{G}(m_1) \\
o_2 & = & \langle c_0, c_1 \rangle \\
m & = & \langle m_0, m_1 \rangle[b]
\end{array}$$

So, the simulator may proceed by picking m_0, m_1 at random on its own, and set $m = \langle m_0, m_1\rangle[b]$ just as in the real execution. However, the simulator cannot compute c_0 and c_1 as in RealR because it does not know m_2'. This is addressed by using the same message m' twice, counting on the pseudorandom masking to hide this deviation from a real protocol execution. Formally, the simulator SimR is defined as follows:

$$\begin{array}{rcl}
\mathsf{SimR}(m', b) & = & (b', m, o_2) \\
b' & = & b \\
m_0 & \leftarrow & \mathbb{M}_n \\
m_1 & \leftarrow & \mathbb{M}_n \\
m & = & \langle m_0, m_1 \rangle[b] \\
c_0 & = & m' \oplus \mathcal{G}(m_0) \\
c_1 & = & m' \oplus \mathcal{G}(m_1) \\
o_2 & = & \langle c_0, c_1 \rangle
\end{array}$$



Combining the simulator with OT_ℓ results in the following ideal system:

$$\begin{array}{rcl}
\mathsf{IdealR}(m_2', b) & = & (m, o_2) \\
m_0 & \leftarrow & \mathbb{M}_n \\
m_1 & \leftarrow & \mathbb{M}_n \\
c_0 & = & m_2'[b] \oplus \mathcal{G}(m_0) \\
c_1 & = & m_2'[b] \oplus \mathcal{G}(m_1) \\
o_2 & = & \langle c_0, c_1 \rangle \\
m & = & \langle m_0, m_1 \rangle[b]
\end{array}$$

As expected, the two systems IdealR, RealR are indistinguishable for both $b = 0$ and $b = 1$. For example, $\mathsf{RealR}(m_2', 0)$ and $\mathsf{IdealR}(m_2', 0)$ are equivalent because they are both computationally indistinguishable from the process that chooses $m \leftarrow \mathbb{M}_n$ and $c \leftarrow \mathbb{M}_\ell$ at random and sets $o_2 = \langle m_2'[0] \oplus \mathcal{G}(m), c\rangle$. The case when $b = 1$ is similar. At this point it would be very tempting to conclude that RealR and IdealR are equivalent, but they are not: they can be easily distinguished by an adversary that sets $m_2' \neq \bot$ and $b = \bot$. In fact, $\mathsf{IdealR}(m_2', \bot) = (\bot, \bot)$, but $\mathsf{RealR}(m_2', \bot) = (\langle c_0, c_1\rangle, \bot)$, where $\langle c_0, c_1 \rangle \neq \bot$. So, IdealR and RealR are not equivalent, and the simulator SimR is not valid. This discrepancy between the ideal and real systems highlights a subtle timing bug in the protocol: in order to carry out the simulation, the transmission of $\langle c_0, c_1\rangle$ should be delayed until after the receiver has selected its bit b. However, this information is not available to the sender, and fixing the protocol requires revising the definition of OT, as we will do in the next section.

Other communication channels We conclude this section with a discussion of other possible communication channels and weaker OT variants that leak some information to the environment. For example, one may replace the perfectly secure communication channel $\mathsf{Net}_{\mathbb{M}}$ with an authenticated channel $\mathsf{AuthNet}_{\mathbb{M}}(i, e_i) = (o, e_o)$ that also takes an input $e_i : \{\bot, \top\}$ and provides an output $e_o : \mathbb{M}$ to the environment. The environment output $e_o = i$ is used to leak the transmitted message as well as the timing information about when the message is transmitted. The environment input e_i is used to allow the environment to delay the transmission of the message $o = e_i!i$ to the receiver.

Similarly, one may consider weaker versions of the OT functionality that leak the input timing information $e_o = (m_2!\top, b!\top)$ to the environment, and allow the environment to delay the OT output $m = e_i!m_2[b]$.

We remark that none of these modifications affect the analysis presented in this section. In particular, considering a perfectly secure communication channel Net only makes our insecurity result stronger. Also, leaking the signal $b!\top$ to the environment does not solve the timing bug in the protocol: in order to fix the bug, the sender needs to delay the transmission of $i_2 = \langle c_0, c_1 \rangle$ until $b > \bot$. So, it is not enough to provide this information to the environment. The timing signal $b!\top$ needs to be provided as an output to the honest sender.

4 OT Length Extension

We have seen that the "standard" definition of Oblivious Transfer is inadequate even to model and analyze a simple OT length-extension protocol. In Figure 3 we provide a revised definition of oblivious transfer that includes an acknowledgment informing the sender of when the receiver has provided her selection bit.

$$\begin{aligned} \mathsf{OT}'_{\mathbb{M}}(m_2, b) &= (a, m) \\ m &= m_2[b] \\ a &= (b > \bot) \end{aligned}$$

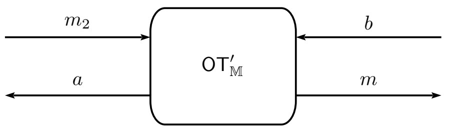
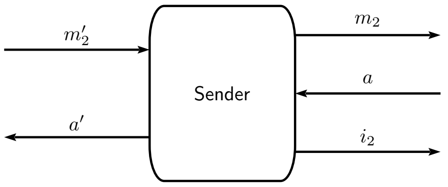


Figure 3: A revised Oblivious Transfer functionality.

We use this revised definition to build and analyze a secure OT length-extension protocol, similar to the one described in the previous section. The OT length extension uses the same $\mathsf{Receiver}$ program as defined in Section 3, but modifies the Sender program by using the signal a to delay the transmission of the message i_2. The new sender also forwards the signal a to the environment to match the new OT' specification:

$$\begin{aligned} \mathsf{Sender}'(m'_2, a) &= (a', m_2, i_2) \\ (m_2, i'_2) &\leftarrow \mathsf{Sender}(m'_2) \\ a' &= a \\ i_2 &= a!i'_2 \end{aligned}$$



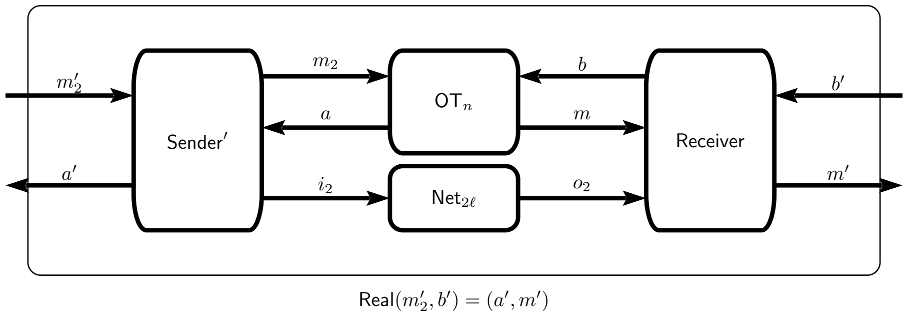
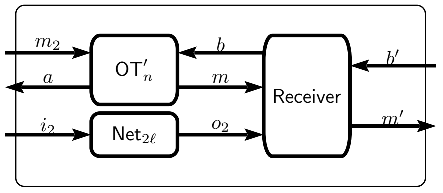
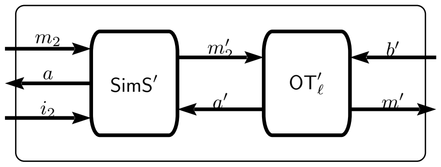


Figure 4: A normal execution of the OT Length Extension protocol.

The Sender and $\mathsf{Receiver}$ programs are interconnected using OT'_n and $\mathsf{Net}_{2\ell}$ as shown in Figure 4. As in the previous section, it is easy to check that the protocol is correct, i.e., combining and simplifying all the equations from the real system in Figure 4 produces a set of equations identical to the revised definition of the ideal functionality $\mathsf{OT}'(m'_2, b') = (a', m')$. Security when the sender is corrupted is also similar to before. The real and ideal systems in this case are given by



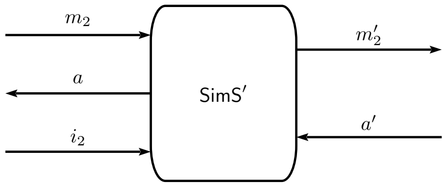
$\mathsf{RealS}(m_2, i_2, b') = (m', a)$



$\mathsf{IdealS}(m_2, i_2, b') = (m', a)$

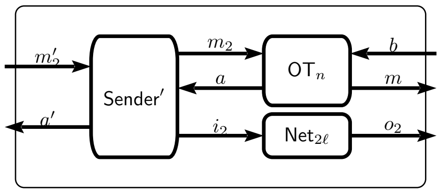
We see that this time SimS' has an additional input a' and output a. We adapt the simulator from the previous section simply by adding an equation that forwards the a' signal from OT' to the external environment:

$$\begin{aligned} \mathsf{SimS}'(m_2, i_2, a') &= (a, m'_2) \\ m'_2 &= \mathsf{SimS}(m_2, i_2) \\ a &= a' \end{aligned}$$

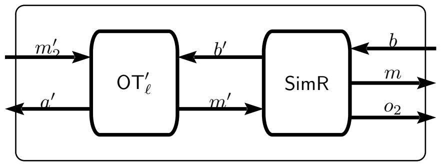


$\mathsf{RealS}(m_2, i_2, b')$ and $\mathsf{Ideal}(m_2, i_2, b')$ are equivalent because they both output $m' = o_2[b'] \oplus \mathcal{G}(m_2[b'])$ and $a = (b' > \perp)$. So, the protocol is still perfectly secure against corrupted senders according to the revised OT' definition.

We now go back to the analysis of security against corrupted receivers. The real and ideal systems are:



$\mathsf{RealR}(m'_2, b) = (a', m, o_2)$



$\mathsf{IdealR}(m'_2, b) = (a', m, o_2)$

No change to the simulator are required: we use exactly the same "candidate" simulator SimR as defined in Section 3. Combining and simplifying the equations, gives the following real and ideal systems:

$$\begin{aligned} \mathsf{RealR}(m'_2, b) &= (a', m, o_2) \\ m_0 &\leftarrow \mathbb{M}_n \\ m_1 &\leftarrow \mathbb{M}_n \\ c_0 &= m'_2[0] \oplus \mathcal{G}(m_0) \\ c_1 &= m'_2[1] \oplus \mathcal{G}(m_1) \\ o_2 &= b!\langle c_0, c_1 \rangle \\ m &= \langle m_0, m_1 \rangle[b] \\ a' &= (b > \perp) \end{aligned} \qquad \begin{aligned} \mathsf{IdealR}(m'_2, b) &= (a', m, o_2) \\ m_0 &\leftarrow \mathbb{M}_n \\ m_1 &\leftarrow \mathbb{M}_n \\ c_0 &= m'_2[b] \oplus \mathcal{G}(m_0) \\ c_1 &= m'_2[b] \oplus \mathcal{G}(m_1) \\ o_2 &= \langle c_0, c_1 \rangle \\ m &= \langle m_0, m_1 \rangle[b] \\ a' &= (b > \perp) \end{aligned}$$

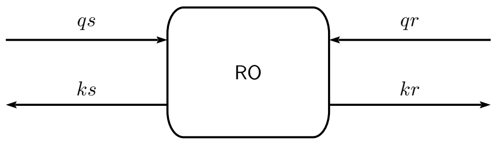
Now, when $b = \perp$, we have $\mathsf{RealR}(m'_2, \perp) = \mathsf{IdealR}(m'_2, \perp) = (\perp, \perp, \perp)$. So, no adversary can distinguish the two systems by not setting b. On the other hand, when $b \neq \perp$, RealR and IdealR are identical to the real and ideal systems from the previous section, augmented with the auxiliary output $a' = (b > \perp) = \top$. As we already observed in Section 3, these two distributions are computationally indistinguishable, proving that the length extension protocol is secure against corrupted receivers.

5 The OT protocol of Chou and Orlandi

In this section we consider the OT protocol proposed by Chou and Orlandi in [CO15]. In the original paper, this is described as a protocol to execute l instances of 1-out-of-m OT, in parallel, i.e., the sender provides an l-dimensional vector of m-tuples of messages, and the receiver (non-adaptively) selects one message from each tuple. For simplicity, we consider the most basic case where $l = 1$ and $m = 2$, i.e., a single OT execution of a basic OT protocol as defined in the previous sections. This is without loss of generality because our results are ultimately negative. So, fixing $l = 1$ and $m = 2$ only makes our results stronger. Our goal is to show that this protocol is not provably secure in equational framework according to a fully asynchronous simulation-based security definition. In order to formally analyze security, we begin by giving a mathematical description of the protocol and model of [CO15] using the equational framework.

The Random Oracle model The protocol of [CO15] is designed and analyzed in the random oracle model [BR93]. So, both parties have access to an ideal functionality RO implementing a random function with appropriately chosen domain Q and range K. Queries from the sender and receiver are answered consistently, and, in general, RO can receive multiple (adaptively chosen) queries from both parties. Formally, the random oracle is modeled by the following functionality, where $f^*(x_1, x_2, \ldots,) = (f(x_1), f(x_2), \ldots)$ is the standard extension of f to sequences:

$$\begin{array}{rcl}
\mathsf{RO}_{Q,K}(qs, qr) & = & (ks, kr) \\
qs, qr & : & Q^* \\
ks, kr & : & K^* \\
f & \leftarrow & [Q \to K] \\
ks & = & f^*(qs) \\
kr & = & f^*(qr)
\end{array}$$



The random oracle starts by picking a function $f : Q \to K$ uniformly at random, and then it uses f to answer any sequence of queries $qs, qr \in Q^*$ from each party. We give separate channels to access RO to the sender (qs) and receiver (qr) to model the fact that random oracle queries are implemented as local computations, and each party is not aware of if/when other players access the oracle. The Sender and $\mathsf{Receiver}$ programs from the protocol of [CO15] only make a small number of queries (two and one respectively.) Moreover, the two sender queries are chosen simultaneously, non-adaptively. So, for simplicity, we restrict $\mathsf{RO}(q_2, q) = (k_2, k)$ to an oracle that receives just a pair of queries $q_2 = \langle q_0, q_1 \rangle \in Q_\perp^{\times 2}$ from the sender and one query $q \in Q_\perp$ from the receiver. We remark that in order to prove security, one should consider an arbitrary (still polynomial) number of (sequential, adaptively chosen) queries to model the adversary/environment ability to compute the RO function locally an arbitrary number of times.[8] However, since our results are negative, fixing the number of queries only makes our result stronger: we show that the protocol is not provably secure even against the restricted class of adversaries that make only this very limited number of random oracle queries.

It has been observed, for example in [CJS14], that a protocol analyzed stand-alone in the traditional random oracle model might lose its security when composed with other instances of protocols in the same random oracle model: either each instance uses an independent random oracle such that the real composed system cannot assume a single hash function, or the composed system suffers from transferability attack. A modified notion called *global random oracle* was proposed in [CJS14] to allow a composed system achieving UC security when all protocols can access a single global random oracle. With respect to this issue, the OT protocol of [CO15] cannot be claimed UC secure and it should be re-defined in the global random oracle model or an equivalent notion. However, such issue is independent of the negative result we are going to present. Since our motivation is to illustrate the use of equational framework in analyzing protocols, for simplicity, we still consider the traditional random oracle model as used in [CO15].

The protocol In order to facilitate a comparison with the original paper, we use as far as possible the same notation as [CO15]. Let $G = \langle B \rangle$ be a group generated by an element B of prime order p. Following [CO15], we use additive group notation, so that the group elements are written as xB for $x = 0, \ldots, p-1$.[9] In

[8]This can be modeled by letting qs and qr range over the set of sequences of queries Q^*, partially ordered according to the prefix ordering relation.

[9]Chou and Orlandi use additive notation to match their efficient implementation based on elliptical curve groups. Here we are not concerned with any specific implementation, but retain the additive notation to match [CO15] and facilitate the

[CO15] it is assumed that group elements have unique, canonical representations (which allows for equality testing), and group membership can be efficiently checked. Here, for simplicity, we assume that all messages representing group elements are syntactically valid, i.e., whenever a program expects a group element from G as input, it will always receive the valid representation of a such a group element (or $\perp$ if the no message has been sent), even when this value is adversarially chosen. This is easily enforced by testing for group membership, and mapping invalid strings to some standard element, e.g., the group generator B.

The protocol uses a random oracle $\mathsf{RO}_{Q,K}(q_2, q) = (k_2, k)$ for functions with domain $Q = G^2 \times G$ and range $K = \{0,1\}^n$, which receives two (parallel) queries $q_2 = \langle q_0, q_1\rangle \in Q^{\times 2}$ from the sender and one query $q \in Q_\perp$ from the receiver.

The protocol also uses a symmetric encryption scheme $(\mathtt{E},\mathtt{D})$, with the same message space $\mathbb{M}_n$ as the OT functionality, and key and ciphertext space $\mathbb{K}_n = \{0,1\}^n_\perp$ equal to the range of the random oracle. The scheme is assumed to satisfy the following properties:

1. Non-committing: There exist PPT algorithms $\mathcal{S}_1$,$\mathcal{S}_2$ such that, for all $m \in \mathbb{M}_n$, the distributions
$$\begin{aligned} \{(e,k) \quad &: \quad k \leftarrow K, e \leftarrow \mathtt{E}(k,m)\} \\ \{(e,k) \quad &: \quad e \leftarrow \mathcal{S}_1, k \leftarrow \mathcal{S}_2(e,m)\} \end{aligned}$$
are identical.[10]

2. Robustness: Let S be a set of keys chosen independently and uniformly at random from $\mathbb{K}_n$. For any PPT algorithms $\mathcal{A}$, if $e \leftarrow \mathcal{A}(S)$, then the set $V_{S,e} = \{k \in S \mid \mathtt{D}(k,e) \neq \perp\}$ of keys under which e can be successfully decrypted has size at most 1 with overwhelming probability (over the choice of S and the randomness of $\mathcal{A}$.)

A simple encryption scheme satisfying these property is given by $\mathtt{E}(m,k) = (m, 0^n) \oplus k$, i.e., padding the message with a string of zeros for redundancy, and masking the result with a one-time pad.

Using this notation, the protocol of [CO15] is described by the following equations, and its execution is depicted in Figure 6.

$$\begin{aligned} \mathsf{Sender}(m_2, k_2, Y) &= (q_2, X, c_2) \\ y &\leftarrow \mathbb{Z}_p^* \\ X &= yB \\ q_2[0] &= ((X,Y), yY) \\ q_2[1] &= ((X,Y), yY - yX) \\ c_2[0] &\leftarrow \mathtt{E}(k_2[0], m_2[0]) \\ c_2[1] &\leftarrow \mathtt{E}(k_2[1], m_2[1]) \end{aligned}$$

$$\begin{aligned} \mathsf{Receiver}(k, X, c_2, b) &= (q, Y, m) \\ x &\leftarrow \mathbb{Z}_p^* \\ Y &= bX + xB \\ q &= ((X,Y), xX) \\ m &= \mathtt{D}(k, c_2[b]) \end{aligned}$$

Figure 5: The OT protocol of Chau and Orlandi.

Here we briefly explain the normal protocol execution: Sender first samples a random group element X and sends it to $\mathsf{Receiver}$; once it receives Y from $\mathsf{Receiver}$, it submits a pair of queries q_2 to RO; and once it receives random keys k_2 from RO, it encrypts messages m_2 under the keys k_2, and it sends the ciphertexts pair c_2 to $\mathsf{Receiver}$. On the other hand, $\mathsf{Receiver}$ firsts samples a random group element xB, and upon receiving X from Sender it computes $Y = bX + xB$ and sends it to Sender; it then submits a query q to RO, and once the random key k and the ciphertexts c_2 are all received, it decrypts $c_2[b]$ using k to get the desired message m.

In the following subsections, we show that this protocol cannot be proved secure, neither according to the classic OT definition given in Figure 1, nor according to our revised OT' definition of Figure 3 that includes

comparison with the original protocol description.

[10]In fact, computational indistinguishability is enough, but it is easy to achieve perfect security.

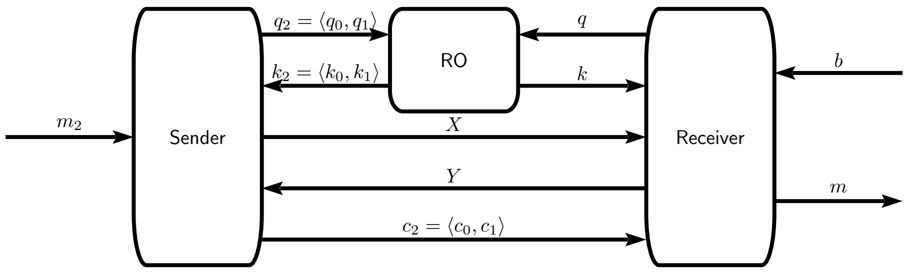
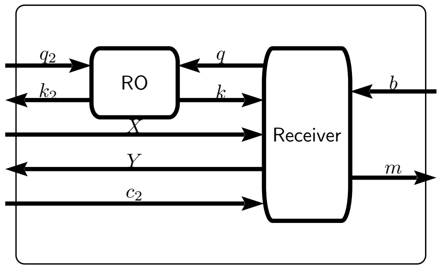

$q_2 = \langle q_0, q_1 \rangle$
RO
q
$k_2 = \langle k_0, k_1 \rangle$
k
b
m_2
Sender
X
Receiver
Y
m
$c_2 = \langle c_0, c_1 \rangle$

Figure 6: A normal execution of the OT protocol of Chou and Orlandi.

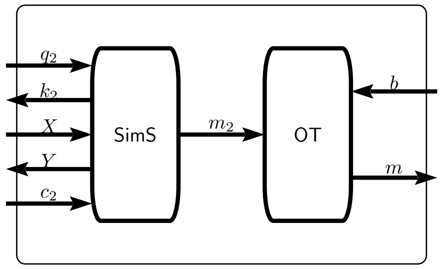
the $a = (b > \bot)$ signal to the sender. Specifically, first, in subsection 5.1 we show that if the definition from Figure 1 is used, then the protocol cannot be proved secure against corrupted senders. This is for reasons very similar to those leading to the failure simulation in Section 3. This suggests that one may fix the problem by considering our revised OT' definition, and suitably modifying the sender program. However, in subsection 5.2 we show that when the revised OT' definition is used, then the protocol cannot be proved secure against corrupted receivers. In addition, in subsection 5.3 we consider a weaker OT'' definition in which the signal $a = (b > \bot)$ is leaked to the environment, instead of being sent to the sender, and we show that the protocol is still not secure in this model when the receiver is corrupted.

5.1 Corrupted sender

We analyze the security of the OT protocol with respect to the standard OT functionality, and consider the case when the sender is corrupted. The corresponding real and ideal systems are shown in the following diagrams:



$\mathsf{RealS}(q_2, X, c_2, b) = (k_2, Y, m)$



$\mathsf{IdealS}(q_2, X, c_2, b) = (k_2, Y, m)$

For the protocol to be secure, the two systems should be computationally indistinguishable (for some simulator program SimS.) We will describe the attacks/environments Env_0 and Env_1, and show that for any simulator SimS, at least one of them distinguishes the real and ideal systems with nonnegligible advantage. We recall that a distinguishing environment connects to all input and output channels of the system, and produces one external output $t \in \{\bot, \top\}$. The distinguishing advantage of the environment Env_x is given by

$$\mathtt{Adv}(\mathsf{Env}_x) = \left|\Pr\{\mathsf{Env}_x[\mathsf{RealS}] = \top\} - \Pr\{\mathsf{Env}_x[\mathsf{IdealS}] = \top\}\right|.$$

The two distinguishers work as follows:

- $\mathsf{Env}_0(k_2, Y, m) = (q_2, X, c_2, b, t)$ sets $q_2 = \bot$, $X = B$, $c_2 = \bot$ and $b = \bot$, and outputs $t = (Y > \bot)$.
- $\mathsf{Env}_1(k_2, Y, m) = (q_2, X, c_2, b, t)$ sets $q_2 = \bot$, $X = B$, $c_2 = \bot$ and $b = 0$, and outputs $t = (Y > \bot)$.

Notice that the only difference between the two environments is in the value of b. Using the equations for the $\mathsf{Receiver}$, we see that in the real system $Y > \perp$ if and only if $b > \perp$. In particular, we have $\Pr\{\mathsf{Env}_0[\mathsf{RealS}] = \top\} = 0$ and $\Pr\{\mathsf{Env}_1[\mathsf{RealS}] = \top\} = 1$. On the other hand, we have

$$\Pr\{\mathsf{Env}_0[\mathsf{IdealS}] = \top\} = \Pr\{\mathsf{Env}_1[\mathsf{IdealS}] = \top\} \tag{4}$$

because when interacting with IdealS, the output value t is independent of b. So, if we let p be the probability in (4), the two environments have advantage $\mathtt{Adv}(\mathsf{Env}_0) = p$ and $\mathtt{Adv}(\mathsf{Env}_1) = 1 - p$. It follows that either Env_0 or Env_1 has distinguishing advantage at least $1/2$.

Intuitively, the protocol is not secure because a corrupted sender (interacting with the real system RealS), learns when the receiver sets $b > \perp$ by observing the incoming message $Y > \perp$. In fact, as we will see, this is the only weakness against corrupted senders, and it can be fixed by augmenting the OT functionality with an output signal $a = (b > \perp)$, similarly to what we did in Section 4.

We remark that securing against corrupted senders with respect to the revised OT' can be proved for adversaries that make an arbitrary number of random oracle queries. Security is demonstrated by the following simulator, which takes the signal $a = (b > \perp)$ as an additional input:

$$\begin{array}{rcl}
\mathsf{SimS}(qs, X, a, c_2) & = & (ks, Y, m_2) \\
f & \leftarrow & [(G^2 \times G) \to K] \\
ks & = & f^*(qs) \\
x & \leftarrow & \mathbb{Z}_p^* \\
Y & = & X!a!xB \\
m_2[0] & = & \sup\{\mathtt{D}(ks[i], c_2[0]) \mid qs[i] = ((X, Y), \ldots)\} \\
m_2[1] & = & \sup\{\mathtt{D}(ks[i], c_2[1]) \mid qs[i] = ((X, Y), \ldots)\}
\end{array}$$

A few explanations about the simulator program are in order. In the definition of the simulator, we have extended the output domain of the decryption function $\mathtt{D}$ with a *top* element $\top$, greater than any other element, so that we can take the supremum $\sup\{x_1, \ldots, x_n\}$ of any set of values. This top element should be interpreted as an error condition that should not occur during any execution (except with negligible probability.) In fact, it follows from the robustness of the encryption scheme that (with overwhelming probability) for any ciphertext $c_2[j]$ there is at most one key $ks[i]$ such that $\mathtt{D}(ks[i], c_2[j]) > \perp$ and taking the sup does not result in $\top$.

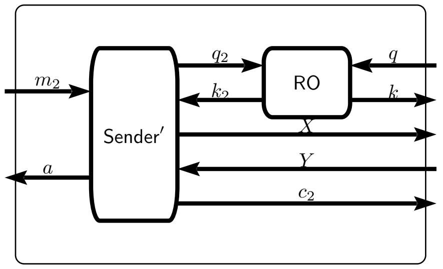
We leave the verification that the simulator is indeed correct to the reader, and move on to the analysis of security against corrupted receivers.

5.2 Corrupted receiver

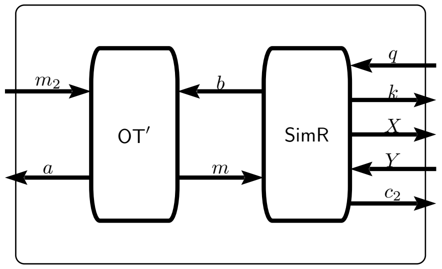
We have seen that when using the standard OT definition, the protocol is not secure against corrupted senders. Here we consider security against the revised OT' definition used to fix the OT length extension protocol in Section 4. Clearly, changing the definition requires also modifying the sender program to output a signal a in order to match the OT' functionality. Since the sender receives only one message (Y) from the receiver, there is only one sensible way to modify the protocol to produce this additional output: setting $a = (Y > \perp)$. Formally, we consider the following modified sender program:

$$\begin{array}{rcl}
\mathsf{Sender}'(m_2, k_2, Y) & = & (a, q_2, X, c_2) \\
(q_2, X, c_2) & \leftarrow & \mathsf{Sender}(m_2, k_2, Y) \\
a & = & (Y > \perp)
\end{array}$$

We leave it to the reader to verify that a real protocol execution $(\mathsf{Sender}' \mid \mathsf{RO} \mid \mathsf{Receiver})\colon (m_2, b) \mapsto (a, m)$ is equivalent to the ideal functionality $\mathsf{OT}'\colon (m_2, b) \mapsto (a, m)$. As we will show in a later section, the protocol is also secure against corrupted senders. So, in order to analyze security we only need to consider the setting where the receiver is corrupted. The real and ideal system in this case are shown in the following diagrams:



$\mathsf{RealR}(m_2, q, Y) = (a, k, X, c_2)$



$\mathsf{IdealR}(m_2, q, Y) = (a, k, X, c_2)$

Security requires that the above two systems are indistinguishable for some simulator program SimR. Unfortunately, as we are about to show, no such simulator exists.

Proposition 1. *For any simulator* SimR*, there is an environment* $\mathsf{Env}(a, k, X, c_2) = (m_2, q, Y, t)$ *(with external output* $t \in \{\bot, \top\}$*) such that the distinguishing advantage*

$$\mathtt{Adv}[\mathsf{Env}] = |\Pr\{\mathsf{Env}[\mathsf{RealR}] = \top\} - \Pr\{\mathsf{Env}[\mathsf{IdealR}] = \top\}|$$

is not negligible.

Proof. Assume there exists a simulator SimR for a corrupted receiver. We build two environments Env_0 and Env_1 as follows, and we will show that $(\mathsf{OT}' \mid \mathsf{SimR})$ can be distinguished from the real system by one of Env_0 and Env_1:

```
$\mathsf{Env}_0(a, k, X, c_2) = (m_2, q, Y, t)$
    sets $d \leftarrow \{0, 1\}$, $x \leftarrow \mathbb{Z}_p^*$, $m_2 = \bot$, $q = \bot$, $Y = dX + xB$,
    outputs $t = (X > \bot \wedge a > \bot)$,
$\mathsf{Env}_1(a, k, X, c_2) = (m_2, q, Y, t)$
    sets $d \leftarrow \{0, 1\}$, $x \leftarrow \mathbb{Z}_p^*$, $m_2 \leftarrow \mathbb{M}_n^{\times 2}$, $q = ((X, Y), xX)$, $Y = dX + xB$,
    outputs $t = (X > \bot \wedge m_2[d] = \mathtt{D}(k, c_2[d]))$.
```

The Sender program always sets $X > \bot$; so the simulator SimR must also set $X > \bot$ on any input with overwhelming probability. Since both Env_0 and Env_1 can distinguish real and ideal systems when SimR sets $X = \bot$, we assume in the following that SimR always sets $X > \bot$, and we show that even such conditional simulator implies a contradiction.

Let u_i be the input distribution of SimR when interacting with Env_i for $i = 0, 1$. For any input distribution $u = (m, q, Y)$ with components m, q, Y, denote using u^m the distribution of m. Similarly we have u^q, u^Y, $\mathsf{SimR}(u)^b$, etc. When interacting with Env_0 the real system sets $a = \top$, so for the ideal system to be indistinguishable from the real system, SimR must output $b \in \{0, 1\}$ with overwhelming probability. Let $p_0 = \Pr(\mathsf{SimR}(u_0)^b > \bot)$ and $p_1 = \Pr(\mathsf{SimR}(u_1)^b > \bot)$; then $1 - p_0$ is negligible. Notice that $u_0^q = \bot$ and $u_0^m = \bot$ with probability 1, and Env_0 and Env_1 set Y according to the same distribution; so $u_0 \leq u_1$. Since SimR is monotone, we have $p_0 \leq p_1$ and thus $1 - p_1$ is also negligible.

For $i = 0, 1$, let $p_i^0 = \Pr(\mathsf{SimR}(u_i)^b = 0)$ and $p_i^1 = \Pr(\mathsf{SimR}(u_i)^b = 1)$. Then $p_0^0 + p_0^1 = p_0$ and $p_1^0 + p_1^1 = p_1$. Since both $\{0\}$ and $\{1\}$ are open sets in $\mathbb{B}$, by monotonicity of SimR we also have $p_0^0 \leq p_1^0$ and $p_0^1 \leq p_1^1$.

Now let us consider the distinguishing advantage of Env_1. It is easy to see that $\Pr(\mathsf{Env}_1[\mathsf{RealR}] = \top) = 1$. When interacting with the ideal system, let $\tilde{b} = \mathsf{SimR}(u_1)^b$ and then we have

$$\begin{aligned}\Pr(\mathsf{Env}_1[\mathsf{IdealR}] = \top) = &\Pr(t = \top \mid \tilde{b} = \bot)\Pr(\tilde{b} = \bot) + \Pr(t = \top \mid \tilde{b} = d)\Pr(\tilde{b} = d) \\ &+ \Pr(t = \top \mid \tilde{b} = 1 - d)\Pr(\tilde{b} = 1 - d),\end{aligned}$$

where $t = (m_2[d] = \mathtt{D}(k, c_2[d]))$. We claim that p_0^{1-d} is negligible. To see this, fix d and consider the case when $\tilde{b} = 1 - d$. By definition of the OT' functionality, $u_1^m = m_2[\tilde{b}]$, and thus SimR does not learn $m_2[d]$.

Since $m_2[d]$ is randomly selected from $\mathbb{M}_n$, SimR cannot do better than randomly guessing and outputting a ciphertext $c_2[d]$ of it. Then $\Pr(t = \top \mid \tilde{b} = 1-d) \le 2^{-n}$, and thus

$$\Pr(\mathsf{Env}_1[\mathsf{IdealR}] = \top) \le (1-p_1) + \Pr(t = \top \mid \tilde{b} = d)p_1^d + 2^{-n}.$$

Since $p_0^{1-d} \le p_1^{1-d} = p_1 - p_1^d$, we have $p_1^d \le p_1 - p_0^{1-d}$. So for $|1 - \Pr(\mathsf{Env}_1[\mathsf{IdealR}] = \top)|$ to be negligible, p_0^{1-d} must be negligible.

However, if p_0^{1-d} is negligible, then with an overwhelming advantage $p_0^d - p_0^{1-d} = p_0 - 2p_0^{1-d}$ we can run $\mathsf{SimR}(\bot, \bot, Y)$ to distinguish two identical distributions $\{xB \mid x \leftarrow \mathbb{Z}_p^*\}$ (when $d = 0$ in Env_0) and $\{X + xB \mid X \in G, x \leftarrow \mathbb{Z}_p^*\}$ (when $d = 1$ in Env_0) of Y, which is a contradiction. So Env_1 can distinguish IdealR from RealR with nonnegligible advantage. $\square$

5.3 A weaker OT definition

So far we have seen that neither the standard OT nor the revised OT' is suitable for defining security of the [CO15] protocol. To capture the security of [CO15] protocol, we attempted to use a weaker definition of the ideal OT functionality, shown in Figure 7, in which the signal about receiving the selection bit b from the sender is always leaked to the environment. Although the modification is insufficient to prove security, we present the analysis as a more involved example of proofs using the equational framework.

$$\begin{aligned} \mathsf{OT}''(m_2, b) &= (a, m) \\ m &= m_2[b] \\ a &= b! \end{aligned}$$

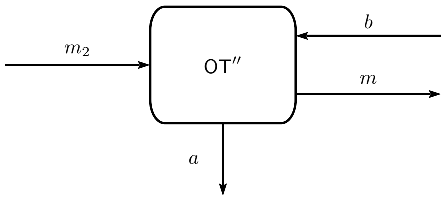


Figure 7: A weaker OT functionality that leaks a to the adversary.

We modify the protocol of Figure 5 by adding a Net functionality between Sender and $\mathsf{Receiver}$ to output a to the environment. The Sender and $\mathsf{Receiver}$ programs are the same as Figure 5. Net is always corrupted and so it can be thought as part of the adversary. We will show later that the modified protocol still is not secure in this weak model; so as in the previous section, we consider the case where the corrupted receiver can make at most one RO query and we will show that there does not exist a simulator for such corrupted receiver. Such restriction only makes our conclusion stronger. This modified protocol can be described by the equations in Figure 8.

$$\begin{aligned} \mathsf{Sender}(m_2, k_2, Y) &= (q_2, X, c_2) \\ y &\leftarrow \mathbb{Z}_p^* \\ X &= yB \\ q_2[0] &= ((X,Y), yY) \\ q_2[1] &= ((X,Y), yY - yX) \\ c_2[0] &\leftarrow \mathtt{E}(k_2[0], m_2[0]) \\ c_2[1] &\leftarrow \mathtt{E}(k_2[1], m_2[1]) \end{aligned}$$

$$\begin{aligned} \mathsf{Receiver}(k, X, c_2, b) &= (q, Y, m) \\ x &\leftarrow \mathbb{Z}_p^* \\ Y &= bX + xB \\ q &= ((X,Y), xX) \\ m &= \mathtt{D}(k, c_2[b]) \end{aligned}$$

$$\begin{aligned} \mathsf{Net}(X, Y) &= (X, Y, a) \\ a &= Y! \end{aligned}$$

Figure 8: An OT'' protocol that always leaks a.

Let us analyze the security of this protocol. When the sender is corrupted, we remove the Sender program and Net functionality from the system, and so the real system is $(\mathsf{RO} \mid \mathsf{Receiver})$. Notice that the definition of $\mathsf{Receiver}$ in the current protocol is the same as in Figure 5, so the real system is secure when the sender is corrupted.

When the receiver is corrupted, we remove the $\mathsf{Receiver}$ and Net from the system and so the real system is $(\mathsf{Sender} \mid \mathsf{RO})$. It turns out that this protocol is not secure when the receiver is corrupted:

Proposition 2. *For the* OT'' *protocol in Figure 8, for any simulator* SimR *of a corrupted receiver, there is an environment* $\mathsf{Env}(k, X, c_2) = (m_2, q, Y, t)$ *such that the distinguishing advantage*

$$\mathtt{Adv}[\mathsf{Env}] = |\Pr\{\mathsf{Env}[\mathsf{RealR}] = \top\} - \Pr\{\mathsf{Env}[\mathsf{IdealR}] = \top\}|$$

is not negligible.

Proof. We define three environments in the following and we will show that, for any SimR, at least one of these environments can distinguish the real system from the ideal system:

$\mathsf{Env}_1(k, X, c_2) = (m_2, q, Y, t)$
  sets $d \leftarrow \{0,1\}$, $x \leftarrow \mathbb{Z}_p^*$, $m_2 = \perp$, $Y = dX + xB$, $q = \perp$,
  outputs $t = (X > \perp \wedge c_2 > \perp)$;
$\mathsf{Env}_2(k, X, c_2) = (m_2, q, Y, t)$
  sets $d \leftarrow \{0,1\}$, $x \leftarrow \mathbb{Z}_p^*$, $m_2 \leftarrow \mathbb{M}_n^{\times 2}$, $Y = dX + xB$, $q = \perp$,
  outputs $t = (X > \perp \wedge c_2 > \perp)$;
$\mathsf{Env}_3(k, X, c_2) = (m_2, q, Y, t)$
  sets $d \leftarrow \{0,1\}$, $x \leftarrow \mathbb{Z}_p^*$, $m_2 \leftarrow \mathbb{M}_n^{\times 2}$, $Y = dX + xB$, $q = ((X, Y), xX)$,
  outputs $t = (X > \perp \wedge m_2[d] = \mathtt{D}(k, c_2[d]))$.

Note that Env_3 is identical to the second environment used in the proof of Proposition 1. As before, the Sender program always sets $X > \perp$. So any simulator SimR must output some $X > \perp$ with overwhelming probability on any input. Similar to Proposition 1, we can assume SimR always output $X > \perp$, and we will show that such conditional simulator implies a contradiction.

Let p_i be the probability that Env_i outputs $t = \top$ when interacting with IdealR, for $i = 1, 2, 3$. The Sender program always sets $X > \perp$. It is also easy to see that, when interacting with Env_1, the Sender program sets $c_2 = \perp$; so p_1 must be negligible. When interacting with Env_2, the Sender program outputs some $c_2 > \perp$; so $1 - p_2$ must be negligible.

Any simulator SimR in IdealR takes an input tuple (m, q, Y, a) and outputs a tuple (X, b, k, c_2). Let u_i be the input distribution to SimR when interacting with Env_i for $i = 1, 2, 3$. Then it is clear that $u_1 \leq u_2 \leq u_3$.

Let $p_b = \Pr(\mathsf{SimR}(u_2)^b = \perp)$. Observe that both Env_1 and Env_2 set $q = \perp$ and they set Y according to the same distribution. If $\mathsf{SimR}(u_2)^b = \perp$, then by monotonicity of SimR we have $\mathsf{SimR}(u_1)^b = \perp$, and thus $u_1^m = u_2^m = \perp$, $u_1^a = u_2^a = \perp$, which implies $u_1 = u_2$. So $\Pr(\mathsf{SimR}(u_2)^b = \perp) \leq \Pr(u_1 = u_2)$. But notice that $\Pr(\mathsf{SimR}(u_1)^{c_2} = \mathsf{SimR}(u_2)^{c_2}) = \Pr(u_1 = u_2)$; thus for $p_1 = \Pr(\mathsf{SimR}(u_1)^{c_2} > \perp)$ and $1 - p_2 = \Pr(\mathsf{SimR}(u_2)^{c_2} = \perp)$ to be both negligible, p_b must be negligible.

However, since $u_2 \leq u_3$, by monotonicity we have $1 - p_b = \Pr(\mathsf{SimR}(u_2)^b > \perp) \leq \Pr(\mathsf{SimR}(u_3)^b > \perp)$. So SimR outputs $b \in \{0,1\}$ with overwhelming probability when interacting with Env_3. Similar to Proposition 1, we can run $\mathsf{SimR}(\perp, \perp, Y, \top)$ to distinguish with overwhelming probability two identical distributions $\{xB\}$ and $\{X + xB\}$ for $X \in G$ and randomly chosen $x \in \mathbb{Z}_p^*$, which is a contradiction. Therefore Env_3 can distinguish two systems, and the proposition follows. □

6 Conclusion

We considered two OT protocols within the equational framework in this paper: The OT length extension protocol and the "simplest" OT protocol by Chou and Orlandi [CO15]. Both examples demonstrated the simplicity and expressive power of the equational framework in analyzing MPC protocols. We found that the traditional formulation of the OT problem does not fit into a fully asynchronous simulation-based security model, and we revised it accordingly to fix it for the OT length extension protocol. Still, the revised formulation does not provide a satisfying security measure for the OT protocol of Chou and Orlandi. As we have shown, the equational framework is useful to carry out rigorous security analysis in a concise way.

For the protocol of [CO15], we would like to find an appropriate OT definition so that the protocol can be proven secure. One possibility is to further modify the OT functionality of Section 5.3 such that it also leaks a signal to the environment when the sender has provided her input messages m_0, m_1. We believe the protocol of [CO15] is provably secure against such OT functionality. But in order to give a formal security analysis we need to upgrade the equational framework to properly model input histories of functions. The details of such development is left to future work.

References

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 186–195, 2004.

[BPW07] Michael Backes, Birgit Pfitzmann, and Michael Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Inf. Comput.*, 205(12):1685–1720, 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145, Oct 2001.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 3–22, 2015.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 61–85, 2007.

[CGT95] Claude Crépeau, Jeroen Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings*, pages 110–123, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical uc security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 597–608, New York, NY, USA, 2014. ACM.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 494–503, New York, NY, USA, 2002. ACM.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology – LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 40–58. Springer International Publishing, 2015.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[GS90] C. A. Gunter and D. S. Scott. Handbook of theoretical computer science (vol. b). chapter Semantic Domains, pages 633–674. MIT Press, Cambridge, MA, USA, 1990.

[HS15] Dennis Hofheinz and Victor Shoup. GNUC: A new universal composability framework. *J. Cryptology*, 28(3):423–508, 2015.

[HUM13] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. Polynomial runtime and composability. *J. Cryptology*, 26(3):375–441, 2013.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 572–591, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In *TCC*, pages 477–498, 2013.

[KT13] Ralf Küsters and Max Tuengerthal. The IITM model: a simple and expressive model for universal composability. *IACR Cryptology ePrint Archive*, 2013:25, 2013.

[Küs06] Ralf Küsters. Simulation-based security with inexhaustible interactive turing machines. In *Computer Security Foundations Workshop, CSFW-19 '06*, pages 309–320. IEEE Computer Society, 2006.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology*, 25(4):680–722, 2011.

[Mau12] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications: Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*, pages 33–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 1–21, 2011.

[MT13] Daniele Micciancio and Stefano Tessaro. An equational approach to secure multi-party computation. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 355–372, New York, NY, USA, 2013. ACM.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer, 1981. Technical Report, TR-81 edn. Aiken Computation Lab, Harvard University.

[Sco82] Dana S. Scott. Domains for denotational semantics. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 577–613, London, UK, UK, 1982. Springer-Verlag.

[SHLG94] Viggo Stoltenberg-Hansen, Ingrid Lindström, and Edward R. Griffor. *Mathematical Theory of Domains*. Cambridge University Press, New York, NY, USA, 1994.

[Wik16] Douglas Wikström. Simplified universal composability framework. In *Theory of Cryptography - TCC 2016-A*, volume 9562 of *LNCS*, pages 566–595. Springer, 2016.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *Foundations of Computer Science, Proceedings of FOCS'86*, pages 162–167. IEEE Computer Society, 1986.